Data-Centric Systems and Applications

Francesco Colace · Massimo De Santo
Vincenzo Moscato · Antonio Picariello
Fabio A. Schreiber · Letizia Tanca
Editors

Data Management in Pervasive Systems

Springer

Data-Centric Systems and Applications

More information about this series at
http://www.springer.com/series/5258

Francesco Colace • Massimo De Santo •
Vincenzo Moscato • Antonio Picariello •
Fabio A. Schreiber • Letizia Tanca
Editors

Data Management in Pervasive Systems

Springer

Editors

Francesco Colace
Università di Salerno
Salerno, Italy

Massimo De Santo
Università di Salerno
Salerno, Italy

Vincenzo Moscato
Università di Napoli
Napoli, Italy

Antonio Picariello
Università di Napoli
Napoli, Italy

Fabio A. Schreiber
Politecnico di Milano
Milano, Italy

Letizia Tanca
Politecnico di Milano
Milano, Italy

ISSN 2197-9723 ISSN 2197-974X (electronic)
Data-Centric Systems and Applications
ISBN 978-3-319-20061-3 ISBN 978-3-319-20062-0 (eBook)
DOI 10.1007/978-3-319-20062-0

Library of Congress Control Number: 2015954106

Springer Cham Heidelberg New York Dordrecht London

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

1 Introduction

The adjective *pervasive* comes from the Latin verb *pervadere*, which literally means *to go through*. However, looking at modern dictionaries, we find wider definitions such as *to permeate* [5] or *to diffuse in order to modify and characterize the atmosphere or the physiognomy of a certain ambient* [3]. The last definition well characterizes modern pervasive information systems, which had a remarkable growth in recent years. Indeed, as shown in Fig. 1, we can find applications of pervasive systems in the most disparate domains, such as health care, archaeology, museums, pollution control, and others: domains which only a few years ago used computers only for administrative purposes and were alien to involve computers directly in their functionality. On the other hand, pervasive systems rest on, and integrate, many different technologies as far as sensing devices, transmission modalities, and networking techniques are concerned.

Back in 1991, Mark Weiser [6] set the essence of modern pervasive systems, stating that they must allow the computer to vanish in the background in the same way as happened with other facilities such as the electric grid or the telephone switching network: as a matter of fact, a typical car has more than a dozen computers and electric motors, but almost no driver is aware of that. This has a fundamental psychological effect, since only when we do not have to concentrate on the individual features of such utilities any more, and they become part of the infrastructure, is our mind free to focus on our goals.

The creation of intelligent pervasive spaces is one of the most interesting opportunities offered by pervasive systems: social and physical ambients can be created with the aid of information and communications technologies (ICT), providing enhanced capabilities for humans to interact with the surrounding environment [4]. These features are useful at home—for instance, for providing services for security, energy management, and water and pollution control or to create assisted-living ambients for impaired or elderly people—but also as proactive and intelligent supports to visiting museums and historical sites.

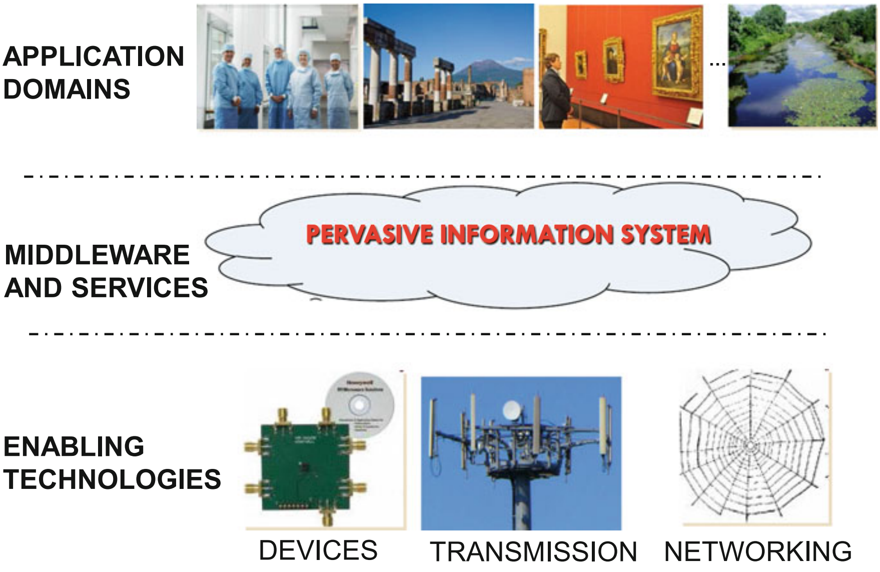


Fig. 1 The components of a pervasive system

Many of these solutions are made possible thanks to the adaptivity and context awareness of pervasive systems; by sensing the environmental conditions, the system dynamically recognizes the situation and context into which it currently operates and behaves accordingly. Adaptivity and context awareness are strictly related to each other and in many real situations are considered as interchangeable; however, while context awareness actually refers to the ability of the system to recognize the current context and to provide, at any time, the necessary contextual information and services [1], adaptivity refers to the execution of behavioral variations in response to changes of context or other parameters that can affect the behavior of the system, even the internal software itself [2]. Therefore, adaptivity and context awareness are complementary in building pervasive applications.

In Fig. 1, the pervasive information system is shown as a layer of middleware and services between the technological level and the application domains.

2 A Guide to Readers

The goal of this book is to provide a systematic description of the plethora of research issues related to the management of information in pervasive systems, illustrating the state of the art in this area. It can be used for a self-contained graduate

(PhD or master) course or for a series of seminars included in other courses on data management or distributed systems.

The book is divided into six parts with a final case study covered in the last part. Part I (Chaps. 1–3) covers very briefly the basic ideas underlying the economical, technological, social, and legal aspects of pervasive systems; Part II (Chaps. 4–7) describes several aspects of sensor networks and data stream processing; Part III (Chaps. 8–10) covers the main aspects of social networks with a special emphasis on cultural heritage applications; Part IV (Chaps. 11–13) describes the personalization and context awareness issues in pervasive environments; Part V (Chaps. 14–16) covers the multimedia aspects, again with a particular attention to the cultural heritage realm. Finally, a real case study is presented in Part VI (Chap. 17) with the description of an application within the DATABENC[1] project of the Campania region in Italy. Each part has its own foreword, guiding the reader through its chapters.

Readers of two different categories can take advantage from reading this book: on one hand, humanities and cultural heritage experts and enthusiasts can be introduced to the enabling technologies that are so promising for their application domain; on the other hand, ICT researchers and professionals can familiarize themselves with the issues of the cultural heritage realm while gaining new knowledge on the advances of pervasive technologies.

As shown in Fig. 2, we suggest the following path to cultural heritage experts. Chapter 1 analyzes and illustrates how new technologies have radically changed cultural and economic models, while Chap. 2 explains the essential technological aspects involved in the implementation and deployment of pervasive information systems. Chapter 3 surveys the main issues related to privacy in emerging pervasive scenarios and discusses some approaches toward their solution. In order to avoid overwhelming non-ICT readers with too many technicalities, we propose to skip some details and sometimes whole chapters: from Chaps. 4, 5, and 7, we recommend to extract only the first two sections. Chapters 8, 10, 11, 12, 13, and 14 are more descriptive and thus can be profitably read by a nontechnologist, while we advise the reader to jump from here directly to Chap. 17, which contains an overall description of the pervasive technologies applied to DATABENC.

The reading path for ICT experts is depicted in Fig. 3. Chapters 1 and 3 provide the readers with the economical, social, and legal aspects raised by the introduction of pervasive technologies, especially in the cultural heritage domain. From here, the readers can probe deeper into one or more technological aspects covering their interests: Part II (sensors, data stream, and storage), Part III (social networks), Part IV (context awareness and personalization), and Part V (multimedia information management). Also the technologists will be interested in the application of all the discussed technologies to the real case of the DATABENC project described in Chap. 17.

[1] High Technology District for Cultural Heritage.

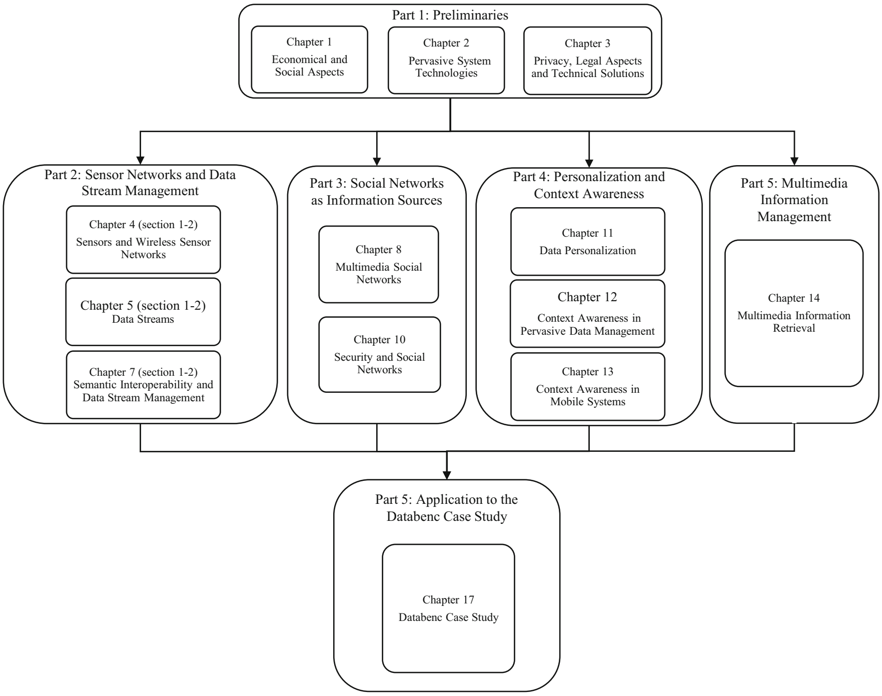

Part 1: Preliminaries
Chapter 1
Economical and Social Aspects
Chapter 2
Pervasive System Technologies
Chapter 3
Privacy, Legal Aspects and Technical Solutions
Part 2: Sensor Networks and Data Stream Management
Chapter 4 (section 1-2)
Sensors and Wireless Sensor Networks
Chapter 5 (section 1-2)
Data Streams
Chapter 7 (section 1-2)
Semantic Interoperability and Data Stream Management
Part 3: Social Networks as Information Sources
Chapter 8
Multimedia Social Networks
Chapter 10
Security and Social Networks
Part 4: Personalization and Context Awareness
Chapter 11
Data Personalization
Chapter 12
Context Awareness in Pervasive Data Management
Chapter 13
Context Awareness in Mobile Systems
Part 5: Multimedia Information Management
Chapter 14
Multimedia Information Retrieval
Part 5: Application to the Databenc Case Study
Chapter 17
Databenc Case Study

Fig. 2 Reading path for humanities and cultural heritage experts

3 Presentation of the Case Study

The DATABENC project is a high-technology district for the management of cultural heritage, recently funded by the Campania region in Italy. Campania boasts one of the largest and most precious cultural heritages in the world: valorizing and promoting such a patrimony, by the adoption of information and communications technologies, is nowadays of paramount importance also at the international level, with a large variety of potential applications.

In particular, DATABENC aims at designing and developing a general framework that provides each cultural site (indoor museums, archaeological sites, historical archives, old town centers, etc.) with several context-aware services for seamlessly assisting users (e.g., visitors or staff personnel) in the exploration and management of the related environment. As in a typical smart-city scenario characterized by the use of Internet-of-Things technologies, the physical sites as well as the users are equipped with all sorts of smart devices and appliances against which the topics of sensor data management, user-originated data operation and reasoning, multimedia and social data management, data analytics and reasoning for event detection and decision making, context modeling and control, automatic data, and service tailoring for personalization and recommendation have to be challenged.

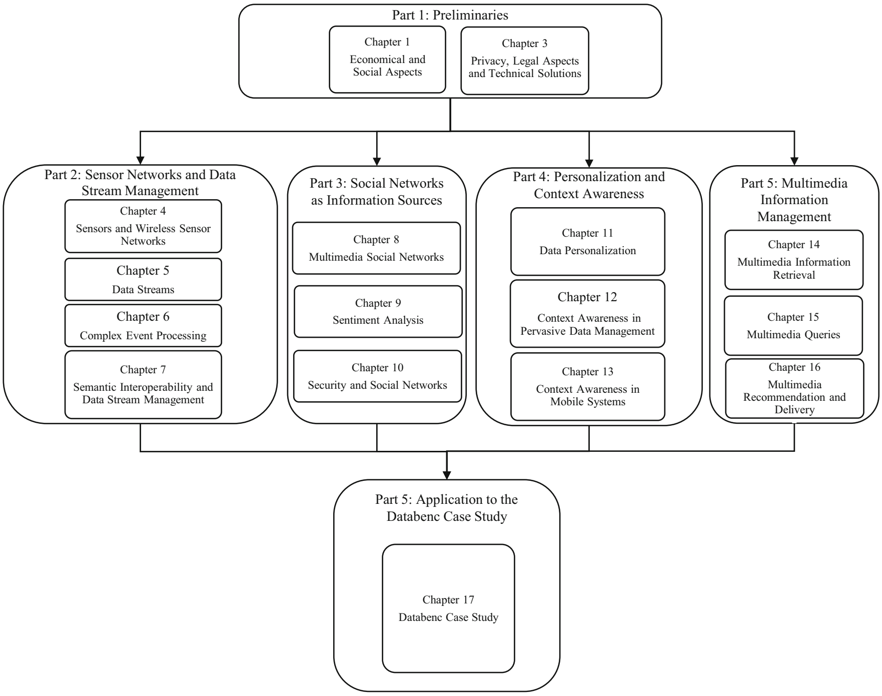

Part 1: Preliminaries
Chapter 1
Economical and Social Aspects
Chapter 3
Privacy, Legal Aspects and Technical Solutions
Part 2: Sensor Networks and Data Stream Management
Chapter 4
Sensors and Wireless Sensor Networks
Chapter 5
Data Streams
Chapter 6
Complex Event Processing
Chapter 7
Semantic Interoperability and Data Stream Management
Part 3: Social Networks as Information Sources
Chapter 8
Multimedia Social Networks
Chapter 9
Sentiment Analysis
Chapter 10
Security and Social Networks
Part 4: Personalization and Context Awareness
Chapter 11
Data Personalization
Chapter 12
Context Awareness in Pervasive Data Management
Chapter 13
Context Awareness in Mobile Systems
Part 5: Multimedia Information Management
Chapter 14
Multimedia Information Retrieval
Chapter 15
Multimedia Queries
Chapter 16
Multimedia Recommendation and Delivery
Part 5: Application to the Databenc Case Study
Chapter 17
Databenc Case Study

Fig. 3 Reading path for ICT experts

As a first motivating example, we can consider tourists who, during their vacation, want to visit a special exhibition of paintings and sculptures offered by a given indoor museum (e.g., the national museum of Capodimonte in Naples). To be considered *smart*, the museum environment should provide users with a set of functionalities for:

- Booking a visit for a specific date and time and buying the related ticket, managing, and user accounting/registration
- During the exhibition visit, accessing appropriate guides which describe the artworks by means of information coming from multiple and heterogeneous multimedia repositories (e.g., Flickr, Panoramio)
- Enabling the objects of the exhibition to "talk," when a user is sufficiently close to them, automatically telling their story by means of multimedia facilities, again according to user preferences and needs
- Analyzing feedbacks, reviews, and comments of other users or experts coming from the most common social networks (Twitter, Facebook, etc.) to have a more detailed vision or an opinion about artworks
- Monitoring the environmental condition of each room by means of sensor networks (e.g., a Wireless Sensor Network (WSN)), for example, detecting some danger and showing the exit in the case of an emergency

- Saving the users' visit experiences in a digital format for future memory, which also allows them to post their comments on social network web sites

As a second example, we can imagine tourists visiting an archaeological outdoor site (e.g., Paestum or Pompeii ruins), endowed with an app which guides and supports them in their visit. In this case, to be considered smart, the environment should provide a set of functionalities for:

- Suggesting useful data and services tailored according to the current user context (user location, user preferences and needs, cultural level, environmental conditions, etc.); as an example, the information can be tailored differently according to the different levels of detail required by an archaeologist or by a nonexpert user
- Dynamically recommending visit paths, using the multimedia description of the cultural attractions or other support information, possibly enabling the publication of comments about user experience on social networks
- Allowing the 3D reconstruction of objects and the interaction between physical and virtual space by means of Virtual Reality technologies
- Monitoring environmental conditions, buildings' state, and users' behavior for security aims

Summarizing, the aim of the DATABENC project is the design and implementation of a Cultural Heritage Pervasive Information System based on the adoption of the future Internet architectural models and technological standards, capable of managing, in an integrated way, sensor-originated data and user-generated content in a pervasive context. Therefore, according to the most recent methodological and technological research on pervasive data management, the system should have the following features:

- It must manage the communication with any kind of sensor that can be deployed in the cultural site of interest (WSN, Radio Frequency Identification (RFID), video cameras, etc.).
- It must provide a set of primitives for the access, retrieval, integration, and analysis of information coming from the different data sources (multimedia repositories, social networks, sensors' database, etc.), managing the correlation with spatial information.
- While supporting data management, it must implement the transformation of the captured data into usable knowledge.
- It must be able to discover and track users within a site using heterogeneous technologies (GPS, Bluetooth, WiFi positioning, etc.).
- It must provide intelligent and personalized access to the knowledge base on the user profile, context state, and applications using context-aware and recommendation facilities.
- It must provide basic primitives for data analytics and reasoning, with the aim of supporting dynamic, on-the-fly personalization and social network analysis applications.

Acknowledgments

The editors would like to thank all the team of the DATABENC project for the discussions and contribution to the design and the implementation of the case study, especially Prof. Angelo Chianese and Dr. Luca Greco.

The chapter "Privacy in Pervasive Systems: Social and Legal Aspects and Technical Solutions" was supported in part by the EC within the 7FP project ABC4EU (grant agreement 312797) and within a H2020 project with grant agreement 644579 and by the Italian Ministry of Research within PRIN project "GenData 2020" (2010RTFWBH).

The chapters "Sensors and Wireless Sensor Networks as Data Sources: Models and Languages," "Data Streams and Data Stream Management Systems and Languages," and "Context Awareness in Pervasive Information Management" were supported by the projects "SHELL" and "GenData 2020" funded by the Italian Government.

The chapter "Applying Semantic Interoperabiltiy Principles to Data Stream Management" has been partially funded by the IBM Faculty Award 2013 granted to Prof. Emanuele Della Valle and by the IBM PhD Fellowship Award 2014/15 granted to Daniele Dell'Aglio.

Salerno, Italy	Francesco Colace
Salerno, Italy	Massimo De Santo
Naples, Italy	Vincenzo Moscato
Naples, Italy	Antonio Picariello
Milan, Italy	Fabio A. Schreiber
Milan, Italy	Letizia Tanca

References

1. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. **36**(4), 19–26 (2007)
2. Cheng, B.H., Al.: Software engineering for self-adaptive systems. In: Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26. Springer, Berlin/Heidelberg (2009)
3. Devoto, G., Oli, G.C.: Il Dizionario della Lingua Italiana, pp. X + 2390. Le Monnier, Firenze (2003)
4. Liu, K.: Pervasive informatics in intelligent spaces for living and working. In: International Conference on Service Operations and Logistics, and Informatics, pp. XVIII–XIX. IEEE, Beijing (2008)
5. Webster, M.: Webster's New Collegiate Dictionary, pp. XXII + 1174. Merriam Webster, Springfield (1961)
6. Weiser, M.: The computer for the 21st century. Sci. Am. **265**(3), 66–75 (1991)

Contents

Part III Social Networks as Information Sources

Part IV Context Awareness and Personalization

Part V Multimedia Information Management

Part VI Application to the DATABENC Case Study

List of Contributors

Massimiliano Albanese George Mason University, Fairfax, VA, USA

Flora Amato University of Naples Federico II, Napoli, Italy

Marco Balduini Politecnico di Milano, Milano, Italy

Jie Bao Microsoft Research Asia, Beijing, China

Ilaria Bartolini University of Bologna, Bologna, Italy

Shi-Kuo Chang University of Pittsburgh, Pittsburgh, PA, USA

Chi-Yin Chow City University of Hong Kong, Kowloon, Hong Kong

Francesco Colace University of Salerno, Fisciano, Italy

Gianpaolo Cugola Politecnico di Milano, Milano, Italy

Sabrina De Capitani di Vimercati University of Milan, Milano, Italy

Aniello De Santo University of Naples Federico II, Napoli, Italy

Massimo De Santo University of Salerno, Fisciano, Italy

Sepideh Deliri George Mason University, Fairfax, VA, USA

Daniele Dell'Aglio Politecnico di Milano, Milano, Italy

Emanuele Della Valle Politecnico di Milano, Milano, Italy

Sara Foresti University of Milan, Milano, Italy

Luca Greco University of Salerno, Fisciano, Italy

Georgia Koutrika HP Labs, Palo Alto, CA, USA

Justin Levandoski Microsoft Research, Redmond, WA, USA

Giovanni Livraga University of Milan, Milano, Italy

Amr Magdy University of Minnesota, Minneapolis, MN, USA

Alessandro Margara USI Universitá della Svizzera Italiana, Lugano, Switzerland

Mohamed F. Mokbel University of Minnesota, Minneapolis, MN, USA

Vincenzo Moscato University of Naples Federico II, Napoli, Italy

Emanuele Panigati Politecnico di Milano, Milano, Italy

Stefano Paraboschi University of Bergamo, Bergamo, Italy

Marco Patella University of Bologna, Bologna, Italy

Ruggero G. Pensa University of Torino, Torino, Italy

Antonio Penta University of Torino, Torino, Italy

Fabio Persia University of Naples Federico II, Napoli, Italy

Antonio Picariello University of Naples Federico II, Napoli, Italy

Silvestro Roberto Poccia University of Naples Federico II, Napoli, Italy

Elisa Quintarelli Politecnico di Milano, Milano, Italy

Emanuele Rabosio Politecnico di Milano, Milano, Italy

Manuel Roveri Politecnico di Milano, Milano, Italy

Pierangela Samarati University of Milan, Milano, Italy

Maria Luisa Sapino University of Torino, Torino, Italy

Mohamed Sarwat Arizona State University, Tempe, AZ, USA

Fabio A. Schreiber Politecnico di Milano, Milano, Italy

V. S. Subrahmanian University of Maryland, College Park, MD, USA

Letizia Tanca Politecnico di Milano, Milano, Italy

Aurelio Tommasetti University of Salerno, Fisciano, Italy

Orlando Troisi University of Salerno, Fisciano, Italy

Massimiliano Vesci University of Salerno, Fisciano, Italy

Carlo Zaniolo UCLA Computer Science Department, Los Angeles, CA, USA

Part I
Preliminaries and Relevant Related Topics

The first part of the book mainly focuses on the foundations of pervasive systems with the related characteristics and research challenges, especially for the cultural heritage management problem.

The new capacities of pervasive and ubiquitous computing are defining new horizons for human creativity and connectivity. This is also because pervasive and ubiquitous computing is very often linked with other emerging technologies, such as semantic knowledge portals, objects in 3D, 2.0 and the semantic web, cloud computing, and open-source software. At the same time, there has been a rapid development in 3G and 4G networks as well as in the use of feature-rich smartphones. This means that everyone can be connected at anytime and anywhere. In the near future, the development of specific hardware as well as software will enable everyone to be in touch with everything and everywhere, thus closing the "circle of pervasiveness." The Internet of the future promises to connect our mobile devices with everything (from the fridge in our homes to special sensors in our cars or even in our bodies), whenever and wherever we are. The Internet of the future, furthermore, will simultaneously generate and make available for everyone huge amounts of data. Scientific and technological research is proceeding so fast nowadays that often a semantic definition (e.g., pervasive and ubiquitous computing) has no time to become established by scholars before some other innovative device emerges in the market. Meanwhile sociocultural styles change and the cost of data management and acquisition is on the decrease. Cloud computing, for example, allows small firms to transfer the cost of hardware and data management to third parties (such as Google, Microsoft, etc.), which makes it possible even for them to have easy access to large amounts of information at low cost. The same is promised with the Internet of Things.

Chapter 1 analyzes and systematizes the abovementioned aspects, and, by means of some examples from the realm of cultural heritage, it shows how new technologies have radically changed cultural and economic models.

Chapter 2 introduces the main architectural issues concerning pervasive systems with the essential technological aspects involved in their implementation and deployment. The authors focus on some typical advanced data management topics that are useful prerequisites for the subsequent chapters such as relational and Not Only SQL (NoSQL) data management, real-time and main memory database management systems, big data, and data analytics.

The globally interconnected society we live in entails an increased exposure of possibly sensitive information and new risks of privacy vulnerabilities. Chapter 3 surveys the main issues related to privacy emerging in pervasive scenarios and discusses some approaches toward their solution.

Chapter 1
The Internet of Things and Value Co-creation in a Service-Dominant Logic Perspective

Aurelio Tommasetti, Massimiliano Vesci, and Orlando Troisi

1.1 Introduction

The new capacities of pervasive and ubiquitous computing are defining new horizons for human creativity and connectivity. This is also because pervasive and ubiquitous computing is very often linked with other emerging technologies, such as the semantic Web, cloud computing, and affective computing. At the same time, there has been a rapid development in 3G and 4G networks as well as in the use of feature-rich smartphones. This means that everyone can be connected at anytime and anywhere. In the near future, the development of specific hardware as well as software will enable everyone to be in touch with everything and everywhere, thus closing the "circle of pervasiveness." The Internet of the future promises to connect our mobile devices with everything (from the fridge in our homes to special sensors in our cars or even in our bodies), whenever and wherever we are. The Internet of the future, furthermore, will simultaneously generate and make available for everyone huge amounts of data. Scientific and technological research is proceeding so fast nowadays that often a semantic definition (e.g., pervasive and ubiquitous computing) has no time to become established by scholars before some other innovative devices flood the market. Meanwhile, sociocultural styles change and the cost of data management and acquisition is on the decrease. Cloud computing, for example, allows small firms to transfer the cost of hardware and data management to third parties (such as Google, Microsoft, etc.), which makes it possible even for them to have easy access to large amounts of information at low cost. The same is promised with the Internet of Things (IoT). The chapter intends to analyze and systematize the abovementioned aspects and, by means of some examples from the realm of

A. Tommasetti (✉) • M. Vesci • O. Troisi
Department of Management & Information Technology, University of Salerno, Salerno, Italy
e-mail: atommasetti@unisa.it; mvesci@unisa.it; otroisi@unisa.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_1

cultural heritage, to illustrate how new technologies have radically changed cultural and economic models. In particular, this chapter aims to point out the impact of IoT on value co-creation, based on a service-dominant logic (S-D logic) perspective.

1.2 Service-Dominant Logic: A Conceptual Framework of Analysis

Service-dominant logic (S-D logic) is a theory that has emerged in the early twenty-first century. It derives from the critical analysis of Vargo and Lusch [46] about the impact of recent nonmanufacturing development on global economies. The contribution of Vargo and Lusch to S-D logic development [47–49] has had a great impact on marketing theory, shifting the root of the economic system from goods to services. Consequently, "the traditional economic worldview, i.e., the goods-dominant logic (G-D logic)" [30], rooted in the general concept of tangible goods or products, has been gradually replaced by a new logic or perspective according to which service is the main driver of current economic development. S-D logic is grounded on ten main foundational premises (FP), which have been developed, modified, and extended by Vargo and Lush between 2004 and 2006. These premises contribute to a better understanding of the pivotal role of service in value exchange, a process open to concrete customer contribution (co-creation), while goods still remain a means for value distribution. These emergent phenomena are summarized in some of the ten premises and, in particular, in FP1 "service is the fundamental basis of exchange," in FP3 "goods are a distribution mechanism for service provision," and in FP6 "the customer is always a co-creator of value." Furthermore, "the terms 'co-creation', 'coproduction', and 'prosumption' refer to situations in which consumers collaborate with firms or with other consumers to produce things of value" [24]. Since the emergence of the S-D logic paradigm, the debate on its specifics, evolution, and relationship with G-D logic has been subject to lively discussion and analysis among scholars. Moreover, this debate has contributed to better define and differentiate the S-D logic theoretical framework, compared to the traditional G-D paradigms.[1] In this respect, the main differences between G-D logic and S-D logic, which are fundamental for this discussion, can be summed up as follows [44, 45, 47]:

- Driver of value
- Resources used
- Customer role

[1] Service-dominant logic "represents a shift in logics of exchange, not just a shift in type of product that is under investigation" [28, 29] leading to a new approach to resources, value (co-)creation, and competition. According to Vargo and Lush, this shift is still in progress, being strictly related to and influenced by the evolution of "information technology (e.g., service-oriented, architecture), human resources (e.g., organizations as learning systems), marketing (e.g., service and relationship marketing, network theory), the theory of the firm (e.g., resource-based theories)" [30].

The first main difference between G-D logic and S-D logic relates to value exchange. In line with the first perspective, it refers to value in exchange, while for the second, it is related to value in use or value in context. “There is no value until an offering is used - experience and perception are essential to value determination.” Another important difference between the two perspectives relates to the resources used. According to G-D logic, the most important resources are the operand ones, and with the emergence of S-D logic on the contrary, operant resources became fundamental in value creation process. Thus, operand resources (e.g., natural resources) are tangible and static and in most cases require action to create value. On the other hand, operant resources (e.g., human skills and knowledge) are intangible and dynamic and capable of acting on operand and other operant resources in order to contribute to value creation [12, 14]. The third fundamental difference between the two perspectives relates to the role that customers play in value creation. G-D logic considers the customer as a mere user or even a destroyer of corporate value, while according to S-D logic, customers participate in value co-creation through the integration of corporate resources and additional ones provided by other subjects, such as public or private companies and even other clients. Thus, co-creation can be considered “the process by which products, services, and experiences are developed jointly by firms and their stakeholders, opening up a whole new world of value. Firms must stop thinking of individuals as mere passive recipients of value, to whom they have traditionally delivered goods, services, and experiences. Instead, firms must seek to engage people as active co-creators of value everywhere in the system” [34].

1.3 The Relevance of the Internet of Things: A Review of Definitions and Main Concepts

The pervasive nature of today’s computing has directly influenced the Internet of Things (IoT) development, making it critical for the future progress of the Internet and the Web 2.0 in terms of people, digital devices, and even common objects’ interconnection and interoperation [13, 18, 41]. From the literature, it emerges that the term “Internet of Things” was used for the very first time in the late 1990s to better define ubiquitous and pervasive computing that “has the potential to change the world, just as the Internet did. Maybe even more so” and that “refers to uniquely identifiable objects (things) and their virtual representations in an Internet-like structure” [5]. It was in 2005 that the International Telecommunication Union (ITU) formally introduced its definition of IoT, according to which it represents “the future of computing and communication, and its development depends on dynamic technical innovation in a number of important fields, from wireless sensors to nanotechnology” [25]. Moreover, “the basic idea of the IoT is that virtually every physical thing in this world can also become a computer that is connected to the Internet”; consequently the spread of these technologies can be considered a radical revolution that led "from anytime and anyplace connectivity for anyone" to a "connectivity for anything." Later, starting from 2008, several studies have pointed out the emergent relationship between IoT and the interoperable communication protocol, even if this relationship and its influence on the whole research field has only been investigated in depth in recent years [8, 10, 17, 22, 27, 35, 50]. Moreover, the European Commission has analyzed the semantic origin of the term which "is composed of two words and concepts: Internet and Things, where Internet can be defined as the world-wide network of interconnected computer networks, based on a standard communication protocol, the Internet suite (TCP/IP), while the term Things refers to objects not specifically identifiable. Therefore, semantically, IoT means a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols" [50]. In the first decade of the twenty-first century, definitions of IoT were mostly based on their logical and operational relationship with "physical objects and beings, as well as virtual data and environments" [40]. In recent years, the IoT has also been considered a "metaphor for the universality of communication processes, for the integration of any kind of digital data and content, for the unique identification of real or virtual objects and for architectures that provide the communicative glue among these components" [35]. According to other perspectives, this concept is related to those things that have "identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts" [11, 50] and to "a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these smart objects over the Internet, query their state and any information associated with them, taking into account security and privacy issues" [22]. Moreover, the Cluster of European Research Projects on the Internet of Things (CERP-IoT)[2] considers IoT as an "integrated part of future Internet and could be defined as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network." According to this definition, smart devices and interfaces contribute to outline real and virtual objects' identities, attributes, and personalities, in order to render them capable of interacting and cooperating in a broadband global network in an effective, practical, and inexpensive way. Consequently, "The IoT allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service" [19], and also because "the basic idea of IoT is the pervasive presence of a variety of things or objects - such as radio-frequency

[2] The Cluster of European Research Projects on the Internet of Things (CERP-IoT) consists of about 30 research initiatives, platforms, and networks dedicated to the identification technologies (e.g., RFID) and to the investigation of the forthcoming Internet-connected and interconnected world of objects.

identification (RFID) tags, sensors, actuators, mobile phones, etc. - which, through unique addressing schemes, interact with each other and cooperate with their neighbors to reach common goals" [17]. Several authors have also investigated the main goal of this emerging paradigm with particular reference to "the ability of smart objects to communicate with each other and build networks of things" [16]. According to Lu and Neng, IoT contributes to the development of things that "have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts."

Currently, the IoT depends on a vast set of properties, related not only to people and objects' active participation, and devices' or smart interfaces accessibility, but also to the interconnection of physical and virtual worlds. The IoT has led to a "virtual world of information technology integrated seamlessly with the real world of things" [43] and "encompasses a variety of technologies and research areas that aim to extend the existing Internet to real world objects" [27]. Currently, debate on IoT is very lively among scholars and practitioners as an increasing number of studies and definitions demonstrate. In order to respond to the lack of standard definitions, the ITU Telecommunication Standardization Sector (ITU-T) has defined IoT "as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based in both existing and evolving interoperable ICTs" [25]. However, the IoT evolution has also been defined in terms of improved network connectivity that goes from "the very simplest electronic devices - to the point where almost anything can connect to the Internet" [21]. In this emerging and advanced world, "connected devices ('temperature sensor') can interact with physical entities ('flowers') by its on device software resources ('sensing software component on sensor device') that can be accessed through standardized services ('Web Service Interface')" [31]. Consequently, "The IoT refers to the emerging trend of augmenting physical objects and devices with sensing, computing, and communication capabilities, connecting them to form a network and making use of the collective effect of the networked objects" [21]. This definition led to an emerging feature of IoT: its pervasiveness. This concept is closely linked to "several technologies and research disciplines that enable the Internet to reach out into the real world of physical objects. Technologies like RFID, short-range wireless communications, real-time localization, and sensor networks are becoming increasingly pervasive, making IoT a reality" [42]. Moreover, this paradigm is related to the possibilities of brand new objects to "become active participants of everyday activities," while "people interact with the technological ecosystem based on smart objects through complex processes. The interactions of these four IoT components, person, intelligent object, technological ecosystem, and process, highlight a systemic and cognitive dimension within security of the IoT" [36]. In conclusion, "IoT does not revolutionize our lives or the field of computing. It is another step in the evolution of the Internet we already have" [32]. In Table 1.1, a classification of the main definitions of IoT has been furnished. The critical analysis

Table 1.1 Definitions and primary components of IoT

Stage	Definition	Author
Machine-to-machine interaction	IoT "refers to uniquely identifiable objects (things) and their virtual representations in an Internet-like structure"	[5]
	IoT, emerging from the investigation of how RFID technologies could participate and contribute "to the networked physical world"	[37]
	The spread of IoT can be considered a radical revolution that led "from anytime and anyplace connectivity for anyone" to a "connectivity for anything"	[25]
	IoT has also been considered a "metaphor for the universality of communication processes, for the integration of any kind of digital data and content, for the unique identification of real or virtual objects and for architectures that provide the communicative glue among these components"	[35]
	Semantically, Internet of Things "means a worldwide network of interconnected objects uniquely addressable, based on standard communication protocols"	[50]
Interconnection of Things	IoT "could be defined as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network"	[8]
	The basic idea of IoT "is the pervasive presence of a variety of things or objects—such as RFID tags, sensors, actuators, mobile phones, etc.—which, through unique addressing schemes, interact with each other and cooperate with their neighbors to reach common goals"	[17]
	IoT refers to "the ability of smart objects to communicate with one another building networks of things"	[16]
	IoT contributes to the development of things that "have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts"	[26]
	Describes a "vision where objects become part of the Internet: where every object is uniquely identified, and accessible to the network, its position and status known, where services and intelligence are added to this expanded Internet, fusing the digital and physical world, ultimately impacting on our professional, personal, and social environments"	[10]

(continued)

Table 1.1 (continued)

Stage	Definition	Author
Web of Things (process based on human-things collaboration)	The IoT led to a "virtual world of information technology integrated seamlessly with the real world of things" [43] and "encompasses a variety of technologies and research areas that aim to extend the existing Internet to real world objects"	[27]
	Connected devices ("temperature sensor") can interact with physical entities ("flowers") by its own device software resources ("sensing software component on sensor device") that can be accessed through standardized services ("Web Service Interface")	[31]
	IoT "as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based in both existing and evolving interoperable ICTs"	[25]
	The Internet of Things "refers to the emerging trend of augmenting physical objects and devices with sensing, computing, and communication capabilities, connecting them to form a network and making use of the collective effect of the networked objects"	[21]
	Several technologies and research disciplines that "enable the Internet to reach out into the real world of physical objects. Technologies like RFID, short-range wireless communications, real-time localization, and sensor networks are becoming increasingly pervasive, making the IoT a reality"	[36]
	The interactions of these four IoT components, person, intelligent object, technological ecosystem, and process, highlight a systemic and cognitive dimension within security of the IoT	[42]
	The IoT does not revolutionize our lives or the field of computing. It is another step in the evolution of the Internet we already have	[32]

of such definitions has contributed to the identification of three main stages of evolution [6, 20].

- Machine-to-Machine interaction (M2M)
- Interconnection of Things
- Web of Things

In particular, the first stage (Machine-to-Machine interaction) is characterized by dynamic information exchange, possibly due to specific interfaces that allow the online connection between machines and devices. In this phase, IoT is based on the idea of "connectivity for anything" [25] and on the growing involvement of RFID technologies [37] contributing to the creation of a worldwide network of interconnected objects. The following stage (Interconnection of Things) is based on the spread of the open Web protocols that lower the barrier to entry for larger number

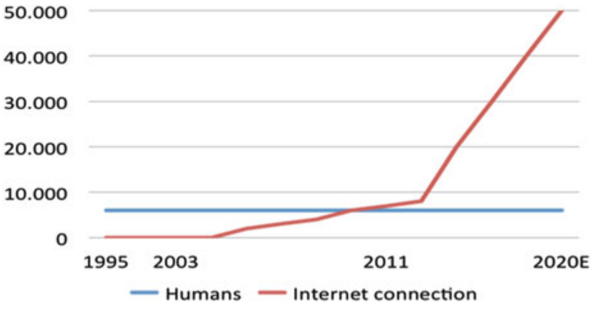


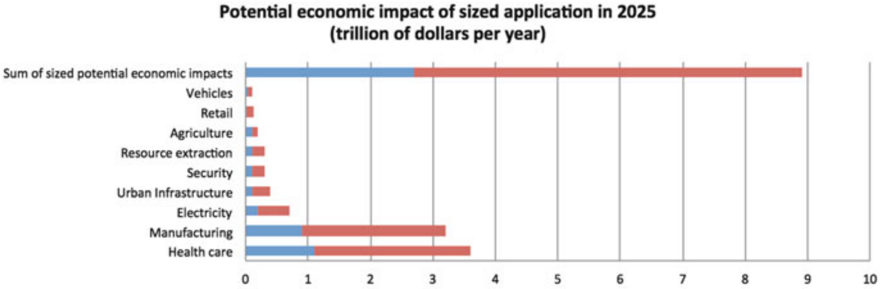
Fig. 1.1 Human beings vs. Internet-connected devices (millions)

of developers, in order to enable the integration of machines and smart objects with current and future business process. In line with the main trends of this stage, IoT represents a dynamic global network [8] made up of a variety of things or objects that interact with each other to reach common goals [16]. The last stage (Web of Things) represents a further vision of IoT based on Web standards that allow the integration in the Internet of everyday life objects containing embedded devices. IoT is now able to provide advanced services by interconnecting (physical and virtual) things, contributing to the emerging trend of augmenting physical objects and devices [21].

1.3.1 The Internet of Things: Trends and Economic Value

Today billions of users are experiencing the IoT paradigm [7]; in fact, the number of Internet-connected devices has surpassed the number of human beings on the planet. The industry predicts that by 2020 possibly 50 billion devices will be connected, a number that is ten times that of all current Internet hosts, including connected mobile phones (see Fig. 1.1).

This enormous number of connected devices shows complex challenges that affect IoT adoption and growth [22]. In a recent study, Cisco has calculated the economic potential of the IoT estimated at $14.4 trillion of value (net profits) for private sector companies globally over the next 10 years. The five main factors that fuel IoT value are as follows: (1) asset utilization (reduced costs)—$2.5 trillion, (2) employee productivity (greater labor efficiencies)—$2.5 trillion, (3) supply chain and logistics (eliminating waste)—$2.7 trillion, (4) customer experience (addition of more customers)—$3.7 trillion, and (5) innovation (reducing time to market)—$3.0 trillion. Similarly, the McKinsey report reveals that various IoT applications could have an economic impact of between $14 trillion and $33 trillion a year by 2025. For this reason, the IoT is regarded as the third and potentially most "disruptive" phase of the Internet revolution after the World Wide Web (the 1990s) and the mobile Internet (the 2000s). The IoT has the potential to create an economic impact of $2.7 trillion to $6.2 trillion annually by 2025 (McKinsey report). The most significant impacts of sized applications would be in health care and manufacturing.



	Health care	Manufacturing	Electricity	Urban Infrastructure	Security	Resource extraction	Agriculture	Retail	Vehicles	Sum of sized potential economic impacts
🟦 From	1.1	0.9	0.2	0.1	0.1	0.1	0.1	0.02	0.05	2.7
🟥 To	2.5	2.3	0.5	0.3	0.2	0.2	0.1	0.1	0.05	6.2

Fig. 1.2 Economic impact of IoT in 2025

Across the health-care applications we analyzed, IoT technology could have an economic impact of $1.1 trillion to $2.5 trillion per year by 2025. Sized applications of the IoT could have a direct economic impact of $2.7 trillion to $6.2 trillion per year in 2025 (see Fig. 1.2).

In particular, data of the Internet-of-Things Observatory at the School of Management, Milan "Politecnico," indicate that performance in the overall IoT Italian market grew by 11 % in 2013, achieving a turnover of approximately 900 million euro. Similarly, objects interconnected via cellular networks—from streetlights to cars, from elevators to slot machines—increased to six million units, up by 20 % compared to 2012. In particular, 47 % of connected objects were topped in the field by Smart Cars. The vast majority (95 %) consisted of box cars with GPS/GPRS tracking applications and applications for the registration of driving parameters for insurance purposes. Following with approximately 1.6 million connected objects (26 %) were Smart Metering for Utilities and Smart Asset Management. Remarkable growth is evidenced particularly as concerns smart electricity and gas meters. With respect to Smart Asset Management in contexts other than those relative to Utilities, data indicate consolidated performance primarily in the monitoring of slot machines and elevators (10 % of the total number of connected objects), while more gradual growth was evidenced in Smart Homes and Buildings (9 % of total objects), Smart Logistics (5 %), and Smart Cities and Smart Environments (2 %).

1.4 From the Internet Revolution to the IoT Evolution

In the previous paragraph, it was highlighted not only how relevant the IoT is for current economic growth but also the increasing involvement of scholars and practitioners in the current debate about the future development of the Internet. Moreover, a growing number of researchers are investigating the possible evolution of the current Internet or the so-called Future Internet, its data integration, and also its relationship with network communication technologies [9]. The emerging concept of Future Internet has been defined as a dynamic global network infrastructure, integrated with many virtual and physical objects, which is based on [9]: (1) standardized communication protocols, (2) Internet of Things, (3) Internet of Media, (4) Internet of Services, and (5) Internet of Enterprises. In this era, IoT represents one of the most disruptive evolutions of daily life, in terms of technologies, accessibility, connectedness/connectivity, communication, and open cooperation. Thus, this phenomenon has led to a paradigm shift according to which "everyday objects become interconnected and smart" [42]. However, IoT has its roots in information and communication technologies (ICT) and in the World Wide Web, two specific phenomena that have definitely changed human communication and interaction. During the latter part of the twentieth century, the spread of the World Wide Web had a critical and revolutionary impact on human society, facilitating real-time information exchange by virtue of a worldwide interconnected web of computers, data, and people [32]. The spread of the Internet had a growing influence on business productivity and efficiency, based as it is on online and real-time information exchange, transmission, storage, and access. The so-called Internet revolution started in the late 1960s, when computer networks enabled the communication between two or more computers [38], while the commercial diffusion of the Web started with the introduction of the TCP/IP. Subsequently, it was only in the 1990s that the Internet became a popular medium. Indeed, it still represents a global channel through which both people and organizations can interact, replacing the traditional mass communication [15, 39]. The spread of this medium was possible also because it has radically changed network organization, in terms of services, costs, and information flows. In particular, network interoperability has made computers and mobile phones interchangeable and interoperable, rendering the global economy more competitive than ever [4]. In the twenty-first century, progress in ICTs has led to a new global world of information economy, in which information represents a critical driver for social and economic growth, innovation, productivity, and job creation [4]. In particular, the emergence of social networking has contributed to enabling users to interact with each other, with other devices or objects over the Internet, and through traditional personal computers or emerging mobile devices. The transition from the traditional Internet to the emerging IoT has passed through the following stages [32]: (1) network creation (based just on the connection of two computers), (2) World Wide Web creation (based on the connection of a large number of computers), (3) mobile Internet (enabling mobile devices to connect to the Internet), (4) mobiles-people-PCs (a complex network in which people join Web via social networks), and (5) Internet of Things (enabling everyday objects to connect to the Internet). In short, IoT plays a pivotal role in the knowledge society and generates new information-based business; if the advent of the Internet has been considered a real and disruptive revolution in communication, in business, and even in everyday life, IoT represents a further evolution of this paradigm, because of its ability to put together several emerging technologies such as radio-frequency identification

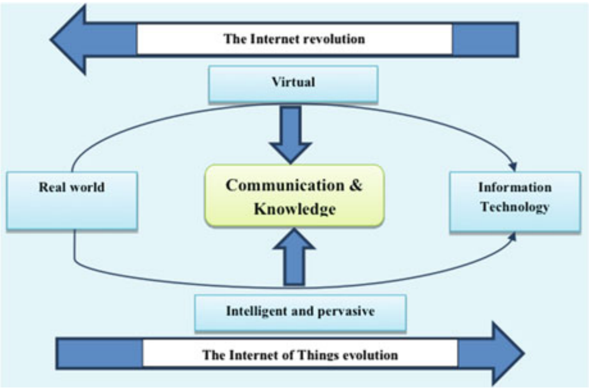

The Internet revolution
Virtual
Real world
Communication & Knowledge
Information Technology
Intelligent and pervasive
The Internet of Things evolution

Fig. 1.3 The Internet revolution vs. the Internet-of-Things evolution

(RFID), near field communication (NFC), and wireless sensor and actuator networks (WSAN). Consequently, it can be assumed that if the Internet revolution has led to the breakthrough of the real world in the virtual one, a phenomenon that has enabled online communication through a worldwide computer network, the recent IoT evolution, based on pervasive and shared intelligence, has enabled people to communicate with smart devices embedded in common objects (Fig. 1.3). In other words, the Internet revolution has contributed to making the real world a liquid and virtual space in which both knowledge and communication flow through the Web. On the other hand, the IoT evolution seems to have defined a reverse stream, according to which a growing number of sensors and devices embedded in spaces and objects of the real world allow users to jump back into a hard world that is real and present to itself again.

IoT now represents the final but definitely not the last stage of a critical evolution that goes from the real world (based on human-to-human interaction) to the World Wide Web that enables people to virtually interact and participate in value creation. Consequently (Fig. 1.4), with the Internet revolution, technology acts as a mediator in value creation. Value creation takes place in a liquid reality, far from the traditional "solid" one. For example, social networks enable humans to interact in a virtual and mediated way, where the real world stays in the background, or rather does not participate in the interaction. On the other hand, as mentioned previously, the IoT evolution has led to a reverse process that is based on the spread of different kinds of sensors placed in the real world. These sensors enable the world to return to its solid state as it seems to be present and real for the users. The IoT evolution has opened a new direction for human-machine interaction; thus this process often takes place in the real world through a growing number of embedded and wireless sensors (GPS, RFID, Micro Controller, System-on-Chip, etc.) that enable the generation of a pervasive intelligence. Users select, interact, and choose the kind of sensor to use and the data to include in value co-creation. The machine-man-machine interaction and the selection of data to use generate knowledge anywhere the user is and renders both the machine and man fundamental elements of the process of co-creation of

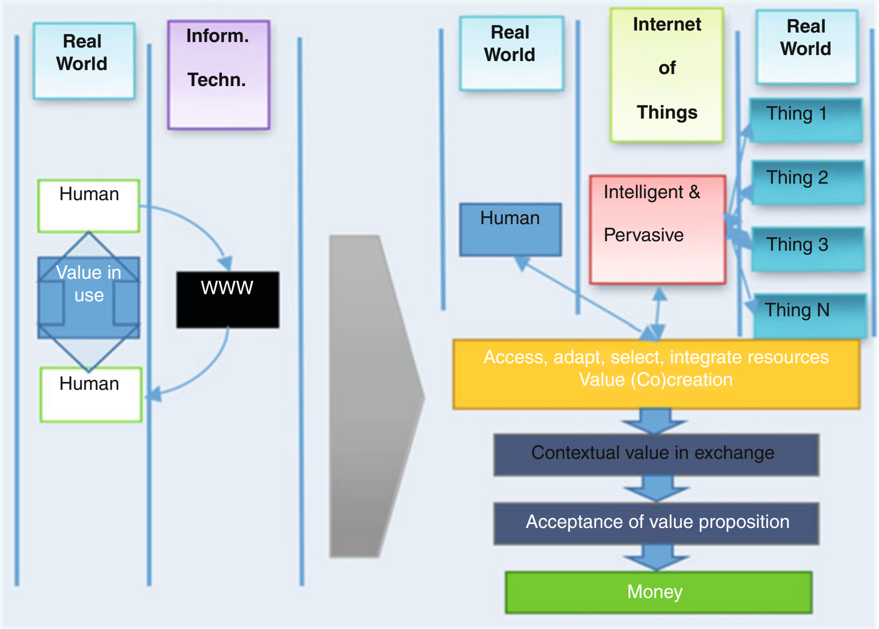


Fig. 1.4 From the Internet of People to the Internet of Things in S-D logic

value. Such value co-creation takes place with use and facilitates acceptance of the value proposition precisely because the latter is co-created and because the user is an active player. The outcome is that wealth production is also facilitated.

IoT has influenced the emergence of two different communication paradigms [6]: (1) human to thing (H2T), according to which people communicate with devices in order to get specific information (e.g., Internet Protocol Television content, file transfer), including remote access to objects by humans, and (2) thing to thing (T2T), according to which objects deliver information (e.g., sensor-related information) to other objects with or without the direct involvement of humans. Clearly, the IoT is not a mature paradigm, and its recent progress is related to [26]: (1) an ubiquitous connectivity dedicated to whenever, whoever, wherever, and whatever type of communication; (2) a pervasive reality creation, in which smart interfaces contribute to a connectable world creation; and (3) ambient assisted intelligence diffusion, which contributes to innovate communication and to increasing value creation. In particular, the IoT vision is based on the facilitation of seamless communications between any kinds of thing or person (e.g., humans and objects), on the rules of the 6Anys (anytime, anywhere, any service, any network, any object, and any human) and 6Cs (connectivity, content, computing, context, collaboration, and cognition), in order to create and deliver innovative services designed for people, business, and society (Fig. 1.5).

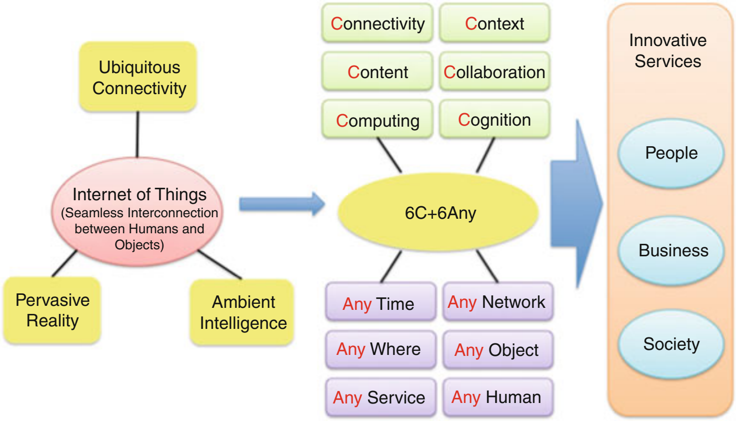


Fig. 1.5 Vision and impact of IoT

Currently, the IoT industry is characterized by the urgent need for interoperability. However, in this context, few specifications and guidelines have been published and others are still under development in order to achieve a future standardization [51].

1.5 The Internet of Things and Value Co-creation: An Introduction in the Cultural Heritage Management

In terms of cultural heritage management, IoT is significant in the following areas [23]:

- Protection—by linking sensors with Internet architecture, it is possible to safeguard and physically protect cultural heritage resources, through the direct and constant monitoring of artifacts.
- Access and interpretation—IoT brings cultural heritage to life, enabling people to better access these resources also by virtue of several additional services and virtual resources (e.g., augmented reality, virtual tour, point of interest, QRCode, etc.).

In this sense, value co-creation is gaining importance and popularity also in the cultural industry, contributing as it does to the capacity of cultural organizations to gain long-lasting competitive advantage and a better response to the emerging demand for personalized and coproduced products, services, and experiences [33]. This represents a minirevolution for the entire industry. In the past, neither visitors nor the public had any influence in service creation or delivery, while nowadays a great number of cultural institutions have opened value creation to direct customer involvement and participation. The emerging pervasive technologies have a direct influence on cultural heritage management, with their aim of creating a virtual environment dedicated to knowledge sharing, creativity, mobility, and accessibility of cultural resources. These technologies and in particular IoT can be considered an operant resource that "is capable of acting on other resources in value creation" [3]. According to S-D logic, as evidenced, two main resources (operant and operand) participate in the value creation process [3, 14]. According to S-D logic, IoT applied to the cultural industry or the so-called Internet of Cultural Things (IoCT) [1, 2] can be considered an operant resource because of its positive influence on customers' access to cultural heritage, on value co-creation, and on the spread of innovative services [3]. The technology also has a direct influence on "the way in which value is determined"; thus when a specific technology is integrated with other resources, "value is uniquely and phenomenologically determined and as technologies are repeatedly combined or integrated with other resources (i.e., innovation occurs), new institutions (e.g., social norms) form" [3]. Furthermore, IoT contributes to the reconfiguration of cultural resources and services, in order to create a new relational network facilitating customer participation in value creation, by virtue of stand-alone interaction with smart objects and embedded sensors. Such technologies enable the access to and the adaption, selection, and integration of cultural resources in a continuous process of value co-creation.

References

1. an Airchinnigh, M.M.: Towards a vision of an Internet of Cultural Things. In: Information Systems & Grid Technologies, Sofia, pp. 1–10, 28–29 May 2009
2. an Airchinnigh, M.M., Strong, G.: Social networks and the national art gallery. In: Publishing in the Networked World: Transforming the Nature of Communication. 14th International Conference on Electronic Publishing (ELPUB), Helsinki, pp. 217–233 (2010)
3. Akaka, M.A., Vargo, S.L.: Technology as an operant resource in service (eco) systems. Inform. Syst. e-Business Manag. **12**(3), 1–18 (2013)
4. Aronson, J.D.: Causes and consequences of the communications and internet revolution. In: The Globalization of World Politics: An Introduction to International Relations, 3rd edn. Oxford University Press, Oxford (2005)
5. Ashton, M.C.: Personality and job performance: the importance of narrow traits. J. Organ. Behav. **19**(3), 289–303 (1998)
6. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
7. Bughin, J., Byers, A.H., Chui, M.: How social technologies are extending the organization. McKinsey Q. **20**(11), 1–10 (2013)
8. CERP-IoT: Internet of things strategic research roadmap. http://www.grifsmonitoring.pdf (2014). Accessed 5 April 2014
9. Chang, K.D., Chang, C.Y., Liao, H.M., Chen, J.L., Chao, H.C.: A Framework for IoT objects management based on future internet IoT-IMS communication platform. In: IMIS 2013, pp. 558–562 (2013)
10. Coetzee, L., Eksteen, J.: The internet of things-promise for the future an introduction. In: IEEE IST-Africa Conference Proceedings, pp. 1–9 (2011)
11. Colace, F., De Santo, M., Greco, L., Chianese, A., Moscato, V., Picariello, A.: Chis: cultural heritage information system. Int. J. Knowl. Soc. Res. **4**(4), 18–26 (2013)

12. Colace, F., Santo, M.D., Greco, L.: An adaptive product configurator based on slow intelligence approach. Int. J. Metadata Semant. Ontol. **9**(2), 128–137 (2014)
13. Colace, F., Santo, M.D., Greco, L.: Simbad: a virtual lab based on the Jini framework. Int. J. Online Eng. **10**(2), 22–29 (2014)
14. Constantin, J.A., Lusch, R.F.: Understanding Resource Management. Irwin Professional, Burr Ridge (1994)
15. Deighton, J.: Commentary on exploring the implications of the internet for consumer marketing? J. Acad. Mark. Sci. **25**(4), 347–351. Las Vegas, Nevada, USA (1997)
16. Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., Schreier, G.: The internet of things for ambient assisted living. In: Seventh International Conference on Information Technology: New Generations, ITNG, pp. 804–809. IEEE, Las Vegas (2010)
17. Giusto, D., Lera, A., Morabito, G., Atzori, L.: The Internet of Things. Springer, Berlin/Heidelberg (2010)
18. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): a vision, architectural elements, and future directions. Futur. Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
19. Guillemin, P., Friess, P.: Internet of things strategic research roadmap. Tech. Rep., The Cluster of European Research Projects. http://www.internet-of-things-research.eu/pdf/IoT (2009). Accessed 6 Sept 2009
20. Guinard, D., Weiss, M., Trifa, V.: Are you energy-efficient? Sense it on the web! In: Adjunct Proceedings of Pervasive, 2009 (7th International Conference on Pervasive Computing), Nara, pp. 1–4 May 2009
21. Guo, B., Zhang, D., Wang, Z., Yu, Z., Zhou, X.: Opportunistic iot: exploring the harmonious interaction between human and the internet of things. J. Netw. Comput. Appl. **36**(6), 1531–1539 (2013)
22. Haller, S., Karnouskos, S., Schroth, C.: The internet of things in an enterprise context. In: Domingue, J., Fensel, D., Tranerso, P. (eds.) Future Internet Systems FIS 2008. Springer, Berlin (2009)
23. Heritage, J.C.: Report on Drivers of Change and the Future of Cultural Heritage. http://www.jpi-culturalheritage.eu/wp-content/uploads/D2.4-Part-1-Report-on-Drivers-of-Change-and-the-Future-of-Cultural-Heritage.pdf (2012)
24. Humphreys, A., Grayson, K.: The intersecting roles of consumer and producer: a critical perspective on coproduction, cocreation and prosumption. Soc. Compass IoT **2**(3), 963–980 (2008)
25. ITU Report: The internet of things. http://www.itu.int/pub/S-POL-IR.IT-2005/e (2005)
26. Lee, G.M., Crespi, N.: Shaping future service environments with the cloud and internet of things: networking challenges and service evolution. In: Leveraging Applications of Formal Methods Verification, and Validation. Springer, Berlin/Heidelberg (2010)
27. Lopez, T.S., Ranasinghe, D.C., Harrison, M., McFarlane, D.: Adding sense to the internet of things-an architecture framework for smart object systems. Pers. Ubiquit. Comput. **16**(3), 1–18 (2011)
28. Lusch, R.F., Vargo, S.L.: Service-dominant logic as a foundation fora general theory. In: The Service-Dominant Logic of Marketing: Dialog, Debate, and Directions, vol. 406. M.E. Sharpe, New York (2006)
29. Lusch, R.F., Vargo, S.L.: Service-dominant logic: reactions, reflections and refinements. Mark. Theory **6**(3), 281–288 (2006)
30. Maglio, P.P., Vargo, S.L., Caswell, N., Spohrer, J.: The service system is the basic abstraction of service science. Inf. Syst. e-Business Manag. **7**(4), 395–406 (2009)
31. Meyer, S., Ruppen, A., Magerkurth, C.: Internet of things-aware process modeling: integrating IoT devices as business process resources. In: Advanced Information Systems Engineering. Springer, Berlin (2013)
32. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutorials **16**(1), 414–454 (2014)
33. Prahalad, C.K., Ramaswamy, V.: Co-creation experiences: the next practice in value creation. J. Interact. Mark. **18**(3), 5–14 (2004)

34. Ramaswamy, V.: Leading the transformation to co-creation of value. Strateg. Leadersh. **37**(2), 32–37 (2009)
35. Report, I.: Casagras/coordination and support action for global rfid-related activities and standardization. http://www.rfidglobal.eu/userfiles/documents/FinalReport.pdf (2008). Accessed 1 April 2014
36. Riahi, A., Natalizio, E., Challal, Y., Mitton, N., Iera, A.: A systemic and cognitive approach for iot security. In: IEEE International Conference on Computing, Networking and Communications (ICNC), pp. 183–188 (2014)
37. Sarma, S., Brock, D.L., Ashton, K.: The networked physical world: proposals for engineering the next generation of computing. IEEE Commer. Autom. Identif. Pervasive Comput. **2**(2), 9–14 (2011)
38. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop Proceedings. First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004, Nottingham, pp. 1–8, 7 September 2004
39. Strauss, J., Frost, R.D.: Marketing on the Internet: Principles of On-line Marketing. Prentice Hall PTR, Upper Saddle River (1999)
40. Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S.: Vision and challenges for realising the Internet of things. In: Cluster of European Research Projects on the Internet of Things (CERP-IoT), March 2010
41. Tan, L., Wang, N.: Future internet: the internet of things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. 376–380 (2010)
42. Trappeniers, L., Feki, M.A., Kawsar, F., Boussard, M.: The internet of things: the next technological revolution. IEEE Comput. **46**(2), 24–25 (2013)
43. Uckelmann, D., Harrison, M., Michahelles, F.: An architectural approach towards the future internet of things. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 1–24. Springer, Heidelberg (2011)
44. Vargo, S.L.: Toward a transcending conceptualization of relationship: a service-dominant logic perspective. J. Bus. Ind. Mark. **24**(5), 373–379 (2009)
45. Vargo, S.L., Akaka, M.A.: Service-dominant logic as a foundation for service science: clarifications. Serv. Sci. **1**(1), 32–41 (2009)
46. Vargo, S.L., Lusch, R.F.: Evolving to a new dominant logic for marketing. J. Mark. **68**(1), 1–17 (2004)
47. Vargo, S.L., Lusch, R.F.: Service-dominant logic: continuing the evolution. J. Acad. Mark. Sci. **36**(1), 1–10 (2008)
48. Vargo, S.L., Lusch, R.F., Morgan, F.W.: Historical perspectives on service-dominant logic. In: The Service-Dominant Logic of Marketing: Dialog, Debate, and Directions, pp. 29–42. M.E. Sharpe, New York (2006)
49. Vargo, S.L., Maglio, P.P., Akaka, M.A.: On value and value co-creation: a service systems and service logic perspective. Eur. Manag. J. **26**(3), 145–152 (2008)
50. Working Group RFID of the ETP EPOSS: Internet of things in 2020: Roadmap for the future. Tech. Rep. http://ec.europa.eu/informationsociety/policy/rfid/documents/iotprague2009.pdf (2008). Accessed 12 June 2014
51. Yang, T.P., Beazley, C., Montgomery, S.B., Dimas, A.S., Gutierrez-Arcelus, M., Stranger B.E., Deloukas, P., Dermitzakis, E.T.: Genevar: a database and java application for the analysis and visualization of snp-gene associations in eqtl studies. Bioinformatics **26**(19), 2474–2476 (2010)

Chapter 2
Pervasive Systems Architecture and the Main Related Technologies

Francesco Colace, Massimo De Santo, Vincenzo Moscato, Antonio Picariello, Fabio A. Schreiber, and Letizia Tanca

2.1 Introduction to Pervasive Data Management

The scope of data management in pervasive systems is to give users an instantaneous and complete access to *any information at anytime and anywhere* in highly dynamic environments where both data and source availabilities vary with high frequency, especially with regard to location and time. In a certain sense, it represents an inverse paradigm with respect to distributed databases: we do not speak anymore of users acting on a passive database but of devices that act both as data producers and consumers, while the system can process the incoming data selecting, for each individual user, the interesting bit of information. Table 2.1, adapted from [13], shows the most relevant features of pervasive information systems compared to those of classical ones.

Figure 2.1 shows a layered architecture for managing data in a pervasive system: in the lowest layer, data are collected from the different sources, which are constituted by directly monitored physical variables as well as information on human activities and artifacts gathered through the Web. In the next layer above it, data manipulation is performed, starting from the integration and storage of information coming from the different sources. The incoming data also affect the state of the context the system operates in, which, in turn, influences the behavior of

F. Colace (✉) • M. De Santo
University of Salerno, Salerno, Italy
e-mail: fcolace@unisa.it; desanto@unisa.it

V. Moscato • A. Picariello
University of Naples Federico II, Naples, Italy
e-mail: vmoscato@unina.it; picus@unina.it

F.A. Schreiber • L. Tanca
Politecnico di Milano, Milan, Italy
e-mail: fabio.schreiber@polimi.it; letizia.tanca@polimi.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_2

Table 2.1 Comparison between classical and pervasive information systems

	Classical IS	Pervasive IS
User	• Committed • Known • Trained Role model: office clerk	• Opportunistic • Unknown • Untrained Role model: citizen
Task	• Specific • Focused on productivity	• Generic • Focused on service delivery and experience
Input	User-generated transactions	• User-generated transactions • Information about context
Access	Desktop workstation	Multiple artifacts
Medium	• Localized/fixed • Homogeneous • *Point and click* paradigm	• Nomadic, spontaneous, temporary Peer to Peer (P2P) networks • Heterogeneous • Multimodal natural interaction
Space	Cybernetic	Physical
Time	Reactive	Proactive
Mobility	Stationary services	Fixed and mobile
Product	Virtual	Virtual and tangible

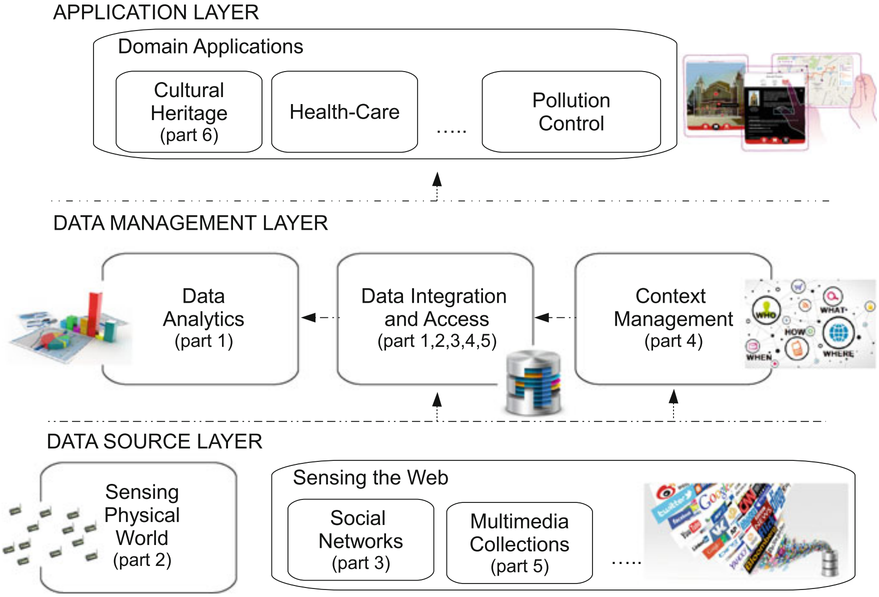


Fig. 2.1 Data management system architecture

the system itself and the tailoring and personalization of the integrated data. At the highest level, the application layer provides services and information to the users of the pervasive system. Each block in the figure carries a reference to the book parts that deal with it.

This chapter summarizes some of the preliminary concepts related to the first two layers; in particular, we briefly describe the main data sources involved in pervasive systems (Sect. 2.2) and summarize the most appropriate data management technologies (Sect. 2.3). Finally, in Sect. 2.4 an introduction to data analytics concepts and techniques is provided.

2.2 Data Sources

Pervasive information systems have to deal with a multitude of heterogeneous data sources, the number and type of which depend on the considered application domain. Pervasive data are mainly the result of two independent phenomena that reached critical mass at the same time: the advent of the *Internet of Things* and the increase in volume of user-generated content produced by *social networks* and smart mobile terminals.

Accordingly, they come from two main sources, sensors and people, and this origin is reflected by their "shape" and possibly recognizable in the form of a (partial, imprecise, hidden) schema characterization. One of the main technological data challenges that are created by pervasive systems, disregarding their context of use, is thus how to understand the intrinsic content of the data, which is typically not provided, as in the traditional databases, by well-organized schematic descriptions: The datasets are expressed in different formats, for instance, relational, extensible markup language (XML), resource description framework (RDF), and even multimedia; when the data are made accessible through services and *wrappers*,[1] the interpretation of the obtained answers may be nontrivial, since the dataset returned as answer is large and unstructured. These issues call for new methods and techniques to identify data regularities and to express them in the form of regular patterns.

As an example, in a Cultural Heritage Pervasive Information System, the data may come from:

- The most common *social networks*, to derive information on users' preferences or moods when they observe a particular cultural item (e.g., a picture or a sculpture) or visit a cultural site (e.g., museum or archeological ruins)
- Different *sensor networks*, possibly deployed in a cultural site, to capture some environmental parameters (e.g., temperature, humidity, etc.) for monitoring aims
- *Digital libraries* or *archives* of cultural organizations to collect all the information describing in detail cultural items that are useful for touristic guide applications

[1] A wrapper is a software that functions as an adapter between a data source and a query system that have two different formats.

- Online *multimedia collections* to retrieve multimedia information (i.e., audio, images, texts, and video) related to cultural items
- Other *Web sources* accessible by means of proper *Web services* (e.g., weather trend, traffic conditions in a given place, hotel reservation info, etc.)

In the following, we briefly describe such common pervasive data sources.

2.2.1 Social Networks

Nowadays, social networks are popular means for sharing information within and among groups of users having similar interests [6]. Every day, social communities like *Facebook*, *Twitter*, *MySpace*, *LinkedIn*, etc., attract millions of users, capturing detailed information about their interests and preferences (useful to derive the related *profile*) or activities (e.g., comments and moods on several topics, events, situations, etc.).

These *social data*, depicting personal relations and activities of users, assume the form of a graph, called *social graph*. In the case of Facebook, the social graph is represented by:

- *Nodes*: basically "things" such as users, photos, pages, or comments
- *Edges*: the connections between things of different or equal nature, such as the photo of a user, a comment on a photo, or the users' friendship relationships
- *Fields*: info about things, such as the birthday of a user or the name of a page

As a matter of fact, in a pervasive context, a social network can be seen as a *social sensing device*, providing the way to access large amounts of social graph information that is hard to obtain by other means [25].

By now, the different social contents can be easily retrieved, exploiting the *application programming interfaces* (APIs) provided by the social network to access the desired information. These APIs are basically functions/routines that can be directly invoked and used by programmers to capture social data useful for building their applications.

As an example, Facebook provides the *graph APIs*.[2] These are particular *representational state transfer* (REST)-compliant APIs that represent the primary way for apps to read and update the Facebook social graph via *hypertext transfer protocol* (HTTP) and can be embedded in several programming languages. Required data are then returned as JavaScript object notation (JSON) packets. In addition, Facebook offers the *Facebook query language* (FQL)[3] to have direct access to social information through proper queries on the tables hosting the data. The

[2] https://developers.facebook.com/docs/graph-api/

[3] https://developers.facebook.com/docs/reference/fql/

type of accessible contents depends on Facebook security policies and on user authorizations and permissions.

2.2.2 Sensor Networks

The increasing need of integrated monitoring systems, especially concerning *safety-critical* applications, caused, in the last decade, a strong development of the related area. The recent advances in information and communication technologies (ICT) and microelectronics have then allowed the production of increasingly effective and cheap *sensors* for the supervision and control of physical and environmental parameters related to places and objects. They can be seen as small-dimension, low-power devices able to sense physical variables such as temperature, pressure, humidity, luminosity, movement, etc., and to communicate, often via wireless, at short distance.

Systems composed by such sensors, with autoaggregation capabilities and low-maintenance requirements, are known as *wireless sensor networks* (WSN) and can be used to instrument environments, possibly accessible from remote clients, for data collection and analysis purposes [28]. These systems, by imposing a tightly connected network of monitoring devices, have the objective, on the one hand, of reducing the need of human intervention and, on the other hand, of providing a valid operating support to decisions for several monitoring problems.

As an example, WSN capabilities can be exploited in a museum to monitor both the temperature and humidity of the rooms and possible displacement of paintings or sculptures.

Various middleware solutions for WSN management have been proposed to access and manage sensor data; they differ in terms of querying and data aggregation models and as to the assumptions about the kind of sensor nodes, topology, size, and other features of the network [10].

Two approaches notable for our purposes are *TinyDB* [18] and *PerLa* [26], which abstract a sensor network as a virtual distributed database system providing a user interface for query formulation and data retrieval.

The user devices themselves, which can communicate the effective user location by using the GPS technology, or a *video surveillance system*, which can detect particular events occurring in a given place (e.g., a person entering a room), can also be considered as a different sort of sensor.

2.2.3 Digital Repositories

Nowadays, the majority of information describing cultural objects can be retrieved from online digital repositories. Indeed, during the last years, several museums, libraries, theatrical foundations, and other types of cultural organizations have realized very interesting digital archives containing the description of their cultural artifacts. As a consequence, different harvesting standards, with metadata specification for the different kinds of objects, have been defined.

As an example, the *lightweight information describing objects*[4] (LIDO) standard has been introduced to support museum items' cataloguing. In particular, LIDO is a harvesting and interchange schema based on XML (*extensible markup language*) intended for delivering object metadata for use in a variety of online services. The strength of LIDO lies in its ability to support the full range of descriptive multilingual information about all kinds of museum items (e.g., art, architecture, cultural history, history of technology, and natural history). Similarly, the *metadata encoding and transmission standard*[5] (METS) is a proposal for encoding descriptive, administrative, and structural metadata regarding objects within a digital library, also expressed using XML. Recently, a European project [5] aims to bridge the gaps between different standards providing a unique solution to metadata specification known as *Europeana data model (EDM) for cultural heritage*.[6]

With the advent of the *semantic Web*, we assisted to a new way of describing resources on the Web: information meaning is explicitly provided, making easier for machines to automatically process and integrate data available on the Web.

In addition to XML-based metadata standard, information—in the Web properly identified by a *unified resource identifier* (URI)—is thus provided together with proper annotation schemes containing a formal description of terminology used for Web documents. To this goal, several standards as *resource description framework* (RDF) and *ontology Web* language (OWL) have been introduced to bridge contents with the related meaning.

Furthermore, the need to connect datasets across the Web has recently led to the birth of the *linked data* movement. It describes a method of publishing structured data so that it can be interlinked and become more useful. It builds upon standard Web technologies such as HTTP, RDF, and URIs but extends them to share information in a way that can be automatically read by computers, enabling data from different sources to be connected and queried. In the case of data accessible to all, we deal with *linked open data* (LOD). The LOD paradigm allows to extend the Web with data commons by publishing various open datasets as RDF on the Web and by setting RDF links among them.

Eventually, multimedia data concerning cultural objects are usually provided with the related description and subjected to copyright policies. Alternatively, open data can be retrieved from image/video-hosting Web site such as *Panoramio*, *Picasa*, *YouTube*, *Facebook*, and *Flickr* and can be accessed by using proper APIs.

[4] http://network.icom.museum/cidoc/working-groups/data-harvesting-and-interchange/what-is-lido/

[5] http://www.loc.gov/standards/mets/

[6] http://pro.europeana.eu/edm-documentation

2.2.4 Web Data Services

Web services are software systems designed to support interoperable machine-to-machine interaction over a network [27] and constitute the simplest way to provide or consume information in a distributed environment.

They refer to the *service-oriented architecture* (SOA) paradigm applied to data sourced from the Web. In particular, Web data services enable *mash-ups*,[7] reuse, and sharing of structured data (such as relational tables), semistructured information (such as XML documents), and unstructured information (such as RSS feeds, content from Web applications, data from online sources).

In a Web data service environment, information can be produced and consumed by applications that can publish and subscribe to it.

Generally, a Web service has an interface described in a machine-understandable format, specifically *Web service description language* (WSDL). Other systems interact with the Web service in a manner prescribed by its description using messages based on the *simple object access protocol* (SOAP), typically conveyed using the HTTP protocol with an XML serialization in conjunction with other Web-related standards [27].

The last-generation *RESTful* Web services do not require XML-based Web service protocols but use only a stateless communication via HTTP with the related primitives of *GET* and *POST* to respectively consume and produce information.

2.3 Advanced Data Management

One of the main storage requirements of the information collected by a *pervasive information system* lies in the need to efficiently manage a huge quantity of data coming from heterogeneous sources (e.g., sensor networks, social networks, digital libraries and archives, multimedia collections, Web pages and services, etc.). The storage systems that manage these data need powerful means to access information quickly, such as *indexes*, i.e., indirect shortcuts pointing to the data searched by users. In fact, like a book, a storage system must also provide some sort of lookup tables used to find the dataset that corresponds to the searched keyword or concept. Besides these basic and classical requirements, the data storage systems supporting pervasive information have to satisfy some specific properties like the fact that, typically, rather than to modifications, they are frequently subject to *append* operations (i.e., addition of new information, like sensor- or Twitter-generated data, or data read from a digital library). This challenge is strictly related to the *Big Data* phenomenon [20], wherein we have to deal with data collections so

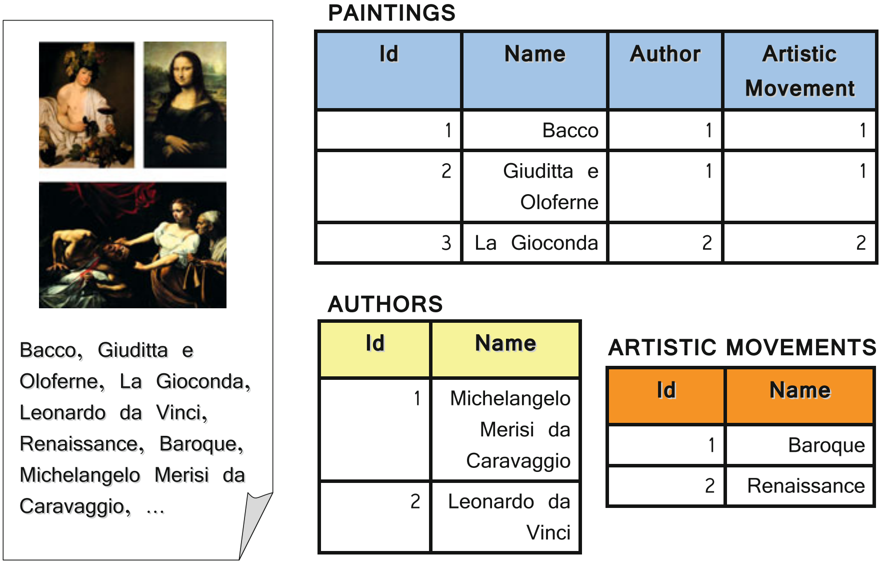
[7] Mash-ups are applications based on the composition of contents and functions that are accessible via the Web.

large and complex that processing them by means of traditional data management systems becomes difficult.

2.3.1 Relational Technologies

In a classical *relational database management system* (RDBMS), data are organized into a fixed set of tables (called *relations*), whose rows (or records), called *tuples*, assume a value for each table column (called *attribute*) and are identified by a *key* value; the relationships between data of different tables are then based on the values of the data themselves. As a result, the information is usually stored in several tables, whose structure (called *schema*) is decided at database design time in such a way as to avoid *data redundancy* and to satisfy proper *constraints*, which depend on the particular application domain. The language used to interact with an RDBMS is the *structured query language (SQL)*. Figure 2.2 shows an example of relational database managing information about famous paintings.

We expect a Cultural Heritage Pervasive Information System to manage not only information about items of interest (paintings, sculptures, etc.) within the given cultural site (museum, archeological ruins, etc.) but also data coming from other pervasive sources such as social data related to user comments, sensor data related to some environmental parameters (e.g., humidity and temperature of rooms), data gathered by Web services about other touristic attractions/accommodations in the


Bacco, Giuditta e Oloferne, La Gioconda, Leonardo da Vinci, Renaissance, Baroque, Michelangelo Merisi da Caravaggio, ...

PAINTINGS

Id	Name	Author	Artistic Movement
1	Bacco	1	1
2	Giuditta e Oloferne	1	1
3	La Gioconda	2	2

AUTHORS

Id	Name
1	Michelangelo Merisi da Caravaggio
2	Leonardo da Vinci

ARTISTIC MOVEMENTS

Id	Name
1	Baroque
2	Renaissance

Fig. 2.2 Example of relational database

same geographic area or meteorological information, and so on. Accordingly, the volume of data to be managed can become extraordinarily large and their nature very heterogeneous; thus, the adoption of a rigid and predefined schema for representing all the possible information does not seem to be the best suited solution [21].

Indeed, relational databases have been designed to be both efficient and reliable with respect to frequent and critical update operations; these requirements generated the need of particular systems that implement the notion of *transaction*. A transaction is a critical portion of an application program wherein database operations are performed and which satisfies four fundamental requirements (the so-called *ACID* properties):

1. *Atomicity*—All the update operations within a transaction must be performed "all or none," that is, if a part of the transaction fails, the entire transaction must fail and the database state must remain unchanged.
2. *Consistency*—Any transaction carried out on a *consistent* database state (i.e., where all the constraints are satisfied) will bring the database into another consistent state.
3. *Isolation*—The concurrent execution of a set of transactions produces the same results as if the same transactions were executed one after the other.
4. *Durability*—If a transaction terminates successfully, its effects are guaranteed to be permanent in the database.

This feature is typical of *online transaction processing* (OLTP) systems, such as banking or commercial information systems, the key goals of which are high throughput for intensive update activities, availability, speed, concurrency, and recoverability. Instead, the write operations carried out in pervasive contexts are typically append-only (as for the progressive storage of sensor data) or complex data retrieval and analytics operations: these, on the one hand, may require long processing times, especially when a query involves a great amount of complex and heterogeneous distributed data, and, on the other hand, do not present such problems as the interference of simultaneous attempts to update the same information. For these reasons, relational technologies are not the most suitable to store and process pervasive data information within tolerable times, and technologies offering high performance and space availability at the cost of a lower reliability may be advisable.

2.3.2 NoSQL Databases

In the last decade, the so-called Not Only SQL (*NoSQL*) technologies have become widespread despite the great availability of commercial and open-source RDBMSs (e.g., *Oracle*, *MySQL*, *PostgreSQL*, to mention the most common ones). NoSQL, for "Not Only SQL," refers to an eclectic and increasingly familiar group of non-relational data management systems where databases are not necessarily stored in tables and generally do not use SQL for data manipulation aims [21]. NoSQL database management systems are very useful when working with a wide amount of data and when their nature does not require the relational model and the ACID properties. The philosophy behind the NoSQL movement is to provide a low-cost (usually the related tools are open source) and very flexible technology for different types of data (social data, sensor data, Web data, etc.) aiming at supporting in an effective and efficient manner the related applications' development. These objectives are essentially achieved facilitating data distribution over *clusters*[8] of computers endowed with large storage capacity and *horizontal scalability*[9] and, at the same time, leveraging a *schemaless* approach and relaxing some of the transaction ACID properties. Indeed, for efficiency and availability purposes, in these systems the data are replicated on several machines, leading to a serious risk of impairing transaction atomicity and of contradictions (*inconsistency*) between two replicas of the same information.

It has been proven (*CAP Theorem* [2]) that only two out of three properties—*c*onsistency, *a*vailability, and tolerance to network *p*artitions—can be simultaneously achieved in distributed computing architectures: for this reason, in these systems the burden of guaranteeing the ACID properties is moved to the application programmers. These considerations must guide the decision about the best technology to support the storage of any system, a decision that has to be taken based on the nature and dynamics of the data at stake.

Summarizing, NoSQL systems are distributed, nonrelational databases designed for large-scale data storage and for massively parallel data processing across a large number of commodity servers [21]. As noted above, this is optimal for pervasive systems, where the large majority of the involved data is typically subject to frequent *append* operations but very seldom to modifications, while large data storage facilities and high computational power are badly needed.

The NoSQL family is characterized by several technologies on the base of the adopted data model [21]. Using the most accepted classification [11], we distinguish for the purposes of this work four main solutions: *key/value stores*, *document stores*, *wide-column stores*, and *graph databases*.

Figure 2.3 shows the complexity of the managed data versus the related size for the different NoSQL technologies. The graph databases are particularly advisable to manage very complex data (e.g., social networks and semantic Web data), while in turn, key/value and column databases are recommended for storing very large data collections.

[8] A computer cluster consists of a set of connected computers that work together so that, in a way, they can be viewed as a single system.

[9] Horizontal scalability is the system's ability to increase its performance when adding new nodes to the related technological infrastructure.

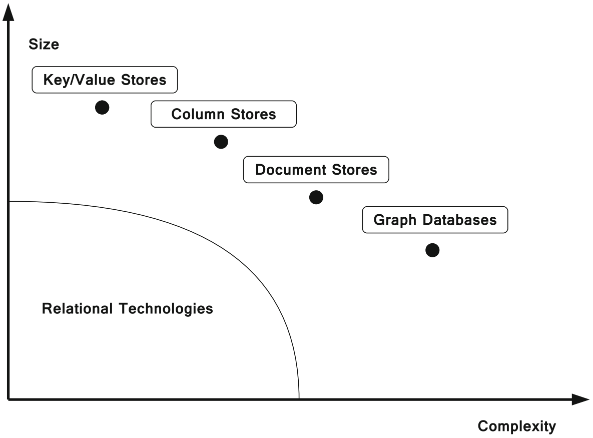
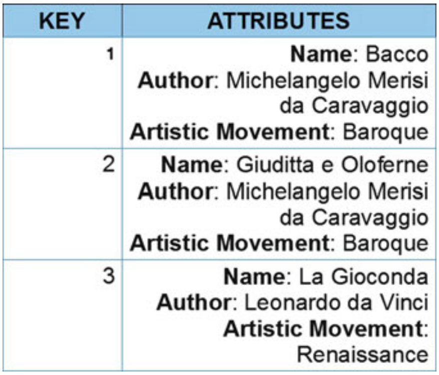


Fig. 2.3 NoSQL: data complexity vs. data size



KEY	ATTRIBUTES
1	**Name**: Bacco **Author**: Michelangelo Merisi da Caravaggio **Artistic Movement**: Baroque
2	**Name**: Giuditta e Oloferne **Author**: Michelangelo Merisi da Caravaggio **Artistic Movement**: Baroque
3	**Name**: La Gioconda **Author**: Leonardo da Vinci **Artistic Movement**: Renaissance

Fig. 2.4 Example of hash table for a key/value database

In this section, we want to provide an overview of the main NoSQL technologies and of how such tools can be useful in meeting data management issues in a pervasive information system.

2.3.2.1 Key/Value Stores

These data management systems typically store information as sequences of *key/value* pairs in simple and stand-alone tables (known as *hash tables*) that can be easily distributed over multiple nodes for managing large amounts of data. In particular, the values may be simple text strings or more complex lists or sets as shown in Fig. 2.4. Data searches are usually performed looking for keys, not values, and are limited to exact matches.

The most famous systems based on this data model are *Dynamo* [3] (implemented by Amazon), *Redis*,[10] *Voldemort*[11] (used in LinkedIn), *Membase*,[12] and *Berkeley DB*.[13]

The simplicity of the key/value technology lies in a lightning-fast, highly scalable retrieval of the values needed for application tasks like managing user profiles/sessions or retrieving item information. The most important issue here is that this information is always added afresh and almost no object is ever updated.

For a Cultural Heritage Pervasive Information System, we can leverage a key/value store to maintain basic information on user profiles and on items of interest (e.g., name, author, and artistic genre of paintings in a museum) for the cultural sites in a given geographic area and, as an example, to quickly provide users with on-a-map overviews of the available resources. Cultural object descriptions can also be directly linked to the related spatial location information within *geographic information systems* (GIS).

2.3.2.2 Document Stores

With such systems, information is stored and managed in the shape of *documents* using an *object-oriented* approach, where each document can be seen as a particular "object" with specific properties.

The majority of these tools accept documents in a variety of forms and encapsulate them in an internal format while extracting some specific *metadata* that are then associated with the document. In particular, documents are usually encoded in standard data exchange formats such as XML, JSON, or BSON (Binary JSON), and data access is realized exploiting proper APIs (application programming interfaces).

Unlike the described key/value stores, a "column" in document databases contains semistructured data and, specifically, attribute name/value pairs (see Fig. 2.5). A single column can host hundreds of such attributes, and the number and type of recorded attributes may vary from row to row (also nested data structures are allowed). Both keys and values are fully searchable in document databases.

The most common document store technologies are *CouchDB*[14] (JSON) and *MongoDB*[15] (BSON).

Document databases are suitable for storing and managing Big Data collections of different documents, like text documents, email messages, multimedia data, XML

[10] http://redis.io/

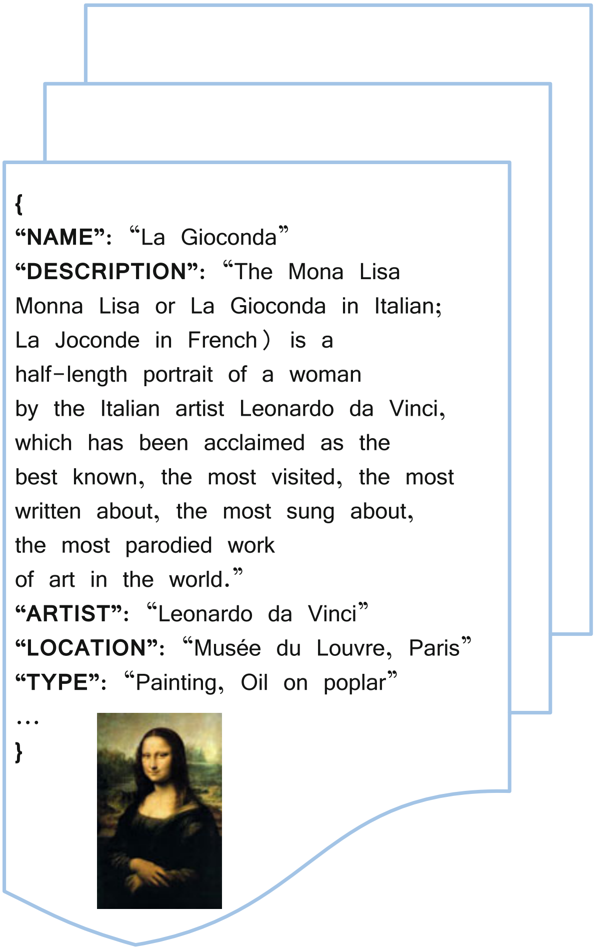
[11] http://www.project-voldemort.com/voldemort/

[12] http://www.couchbase.com/press-releases/membase-general-availability

[13] http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

[14] http://couchdb.apache.org/

[15] http://www.mongodb.org/



```
{
"NAME": "La Gioconda"
"DESCRIPTION": "The Mona Lisa
Monna Lisa or La Gioconda in Italian;
La Joconde in French) is a
half-length portrait of a woman
by the Italian artist Leonardo da Vinci,
which has been acclaimed as the
best known, the most visited, the most
written about, the most sung about,
the most parodied work
of art in the world."
"ARTIST": "Leonardo da Vinci"
"LOCATION": "Musée du Louvre, Paris"
"TYPE": "Painting, Oil on poplar"
...
}
```

Fig. 2.5 Example of information managed by a document store

documents, and so on. They are also good for storing *sparse* data, in other terms irregular (semistructured) data that would require an extensive use of the so-called *null* values (the placeholders for missing information in RDBMSs).

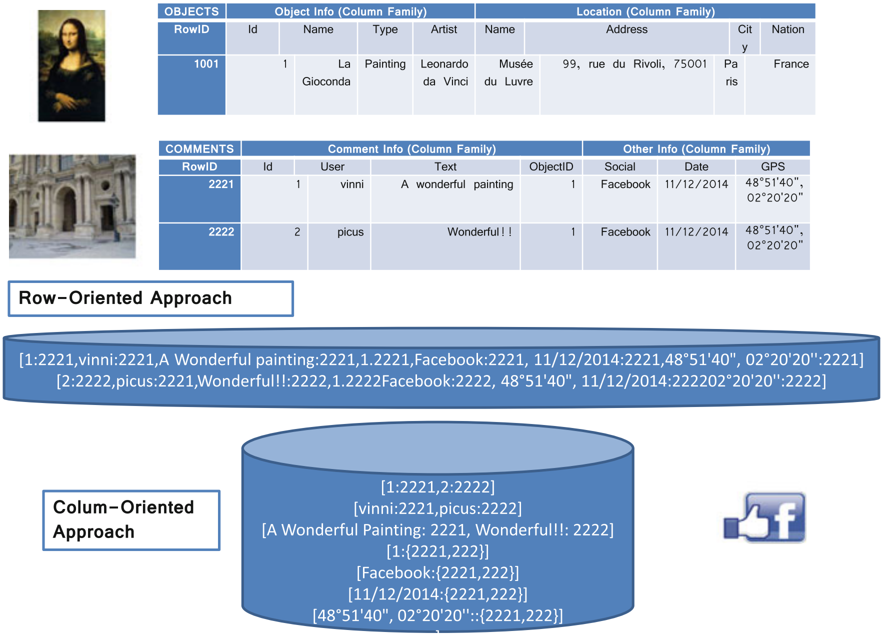
In the case of a Cultural Heritage Pervasive Information System, a document store is well suited to collect and manage the XML documents (e.g., provided by libraries, museums, archives, and audiovisual sectors) containing metadata specification (records) of cultural heritage objects according to the most accepted standards (e.g., LIDO for museums, METS for digital libraries, or EDM of *Europeana* Project [5]).

Multimedia materials related to a cultural object are usually managed directly by the content provider, and the related access is regulated by copyright management policies.

2.3.2.3 Wide-Column Stores

Wide-column databases store information in data tables as “sections” of *columns* of data rather than as rows of data by means of a sequence of key/value pairs, as shown in Fig. 2.6 for the table *Comments*. The column-store approach allows both data compression (i.e., if a given column value is present several times, it can be stored once) and efficient access when it involves particular data scanning (e.g., when an aggregate needs to be computed over many rows but only for a notably small subset of columns). The column-oriented data structure can also accommodate multiple attributes per key (*column family*).

While some column stores have an effective key/value storing engine, the majority is patterned after Google’s *Bigtable*,[16] the petabyte-scale internal distributed data storage system of Google. These generally replicate not just Google’s Bigtable



OBJECTS	Object Info (Column Family)				Location (Column Family)			
RowID	Id	Name	Type	Artist	Name	Address	City	Nation
1001	1	La Gioconda	Painting	Leonardo da Vinci	Musée du Luvre	99, rue du Rivoli, 75001	Paris	France

COMMENTS	Comment Info (Column Family)				Other Info (Column Family)		
RowID	Id	User	Text	ObjectID	Social	Date	GPS
2221	1	vinni	A wonderful painting	1	Facebook	11/12/2014	48°51'40", 02°20'20"
2222	2	picus	Wonderful ! !	1	Facebook	11/12/2014	48°51'40", 02°20'20"

Row-Oriented Approach

[1:2221,vinni:2221,A Wonderful painting:2221,1.2221,Facebook:2221, 11/12/2014:2221,48°51'40", 02°20'20'':2221]
[2:2222,picus:2221,Wonderful!!:2222,1.2222Facebook:2222, 48°51'40", 11/12/2014:222202°20'20'':2222]

Colum-Oriented Approach

[1:2221,2:2222]
[vinni:2221,picus:2222]
[A Wonderful Painting: 2221, Wonderful!!: 2222]
[1:{2221,222}]
[Facebook:{2221,222}]
[11/12/2014:{2221,222}]
[48°51'40", 02°20'20''::{2221,222}]

Fig. 2.6 Example of a column store

[16] http://research.google.com/archive/bigtable.html

data storage structure but Google's distributed file system (GFS) and *MapReduce*[17] parallel processing framework, as is the case with *Hadoop*, which comprises the Hadoop file system (HDFS, based on GFS). The most famous tools that exploit such data model are *HBase*[18] (Bigtable + Hadoop) and *Cassandra* (Dynamo)[19]; they are able to manage billions of rows *times* millions of columns atop clusters of commodity hardware.

The column-store technology is thus particularly suitable to store huge amounts of data over distributed multiple nodes, ensuring quasi-real-time access to data, especially for processing and analytics goals.

In the case of a Cultural Heritage Pervasive Information System, a column database can be used to gather and store all users' comments from the different social networks about cultural heritage objects or data from sensors deployed within a cultural site to support, for example, analytics applications such as sentiment analysis, environmental monitoring, and so on.

2.3.2.4 Graph Databases

Graph databases replace relational tables with graphs of interconnected key/value pairs. In particular, graphs are represented as a set of nodes (objects), edges (objects' relationships), and properties (object and relationship attributes expressed as key/value pairs) as shown in Fig. 2.7. They are focused on the visual representation of information and are thus more user friendly than other NoSQL databases.

Graph database examples are *Neo4j*,[20] *InfoGrid*,[21] and *AllegroGraph*.[22]

In general, graph databases are useful when we are more interested in relationships among data than in the data itself; they are optimized for relationship traversing and not for generic querying.

RDF Stores (e.g., *Sesame* and *Virtuoso*) can be considered as a kind of graph database, where information is stored in the shape of a set of RDF triples (easily representable as a graph). These systems are designed for semantic Web data and provide a specific query language, SPARQL, for retrieving aims.

In a Cultural Heritage Pervasive Information System, graph database technologies can be profitably adopted to manage and represent the relationships between users or between user activities and sensor measures, for instance, with the aim of providing recommendations.

[17] MapReduce is a parallel programming model for processing large datasets with a distributed algorithm on a computer cluster.

[18] http://hbase.apache.org/

[19] http://cassandra.apache.org/

[20] http://neo4j.com/

[21] http://infogrid.org/trac/

[22] http://franz.com/agraph/allegrograph/

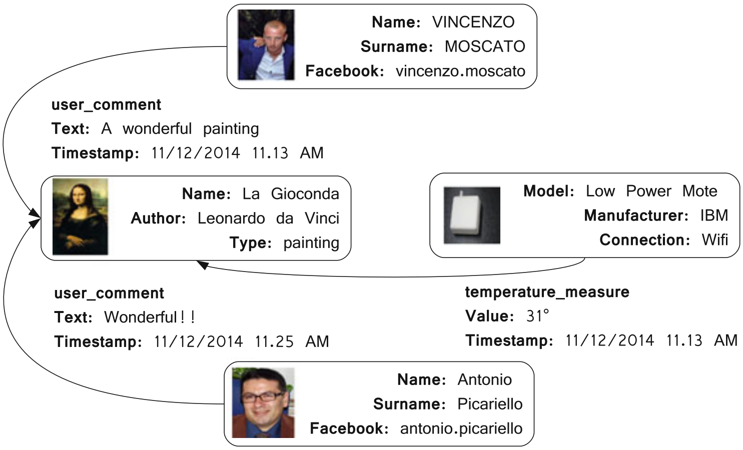


Fig. 2.7 Example of a graph database

2.3.3 *Real-Time Data Management*

The presence of sensors among the data sources in a pervasive information system and the possible need of reaction times strictly dependent on the applications' needs often require the capability of *real-time data management*.

Real-time database management systems (RTDBMS) are systems that manage *time-constrained data* and *time-constrained transactions* in order to predictably meet the applications' needs within given deadlines. RTDBMS are not just DBMSs with a very fast response time—even if speed is often a sufficient but not necessary condition—nor do they explicitly deal with issues related to time representation, such as in temporal databases, even if they are concerned with data temporal validity [9, 29].

Therefore, beyond the functionalities required by an OLTP system, there are two main issues that must be dealt with by an RTDBMS: (1) to guarantee *data temporal consistency* and (2) to guarantee *transaction temporal consistency*. These requirements might conflict with the ACID properties of OLTP systems, and therefore special techniques and algorithms have been proposed for concurrency control and for failure recovery problems [16, 17, 23, 33, 34], which privilege transaction commitment within the deadlines at the expenses of accepting limited imprecisions and approximations in answers.

As to temporal data validity, *absolute* lifetimes must be explicitly defined for each data type since it must reflect the true state of the world; however, when more data types are involved in a transaction, *relative temporal data consistency* must also be considered. As an example, in the environment control system for the rooms of an art gallery, the temperature and humidity must be monitored in order to avoid damage to the exhibited paintings. The time constants of the phenomena being given, the validity of temperature and humidity data can be fixed in 15 min (absolute

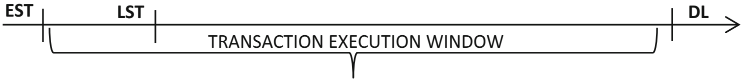


Fig. 2.8 The time line of a transaction

validity), but a further constraint asks that the last update times of the two variables be no more than 5 min apart (relative consistency) in order to reflect a compatible situation.

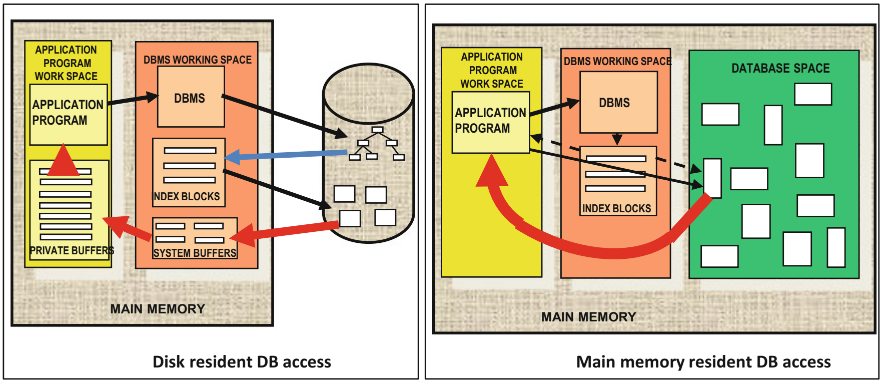
On the other hand, transactions must respect some absolute time milestones, as shown in Fig. 2.8; they cannot start before an earliest start time (EST), because possibly the needed data are not yet ready, and shall start within a latest start time (LST), because otherwise they are not guaranteed to commit within the final deadline (DL). Also periodic invocation can be foreseen with a period *T*, in which case the time line of the transaction must be entirely contained within *T*. These time milestones directly result from the temporal consistency requirements as shown by the environment control system example.

The level at which timing constraints are enforced depends on the *application policy*: (1) violation of *hard constraints* can entail a disaster in safety-critical applications or in control and command systems; (2) with *firm constraints* the transaction loses all of its value at the expiration of the deadline, as in the case of financial transactions; (3) *soft constraints* imply that the transaction results are still valuable for some time after the deadline expired. In order to respect the deadlines, transaction execution time must be *predictable*; therefore, all the data structures and programming practices that can pose a threat to a deterministic execution time, such as recursive and dynamic data structures, dynamic paging, and the dependence of the execution sequence from data values, must be avoided. Also too many disk I/O operations should be avoided owing to nondeterministic latency times, and this is a reason for coupling RTDBMS with main memory data management systems, as we shall see in Sect. 2.3.4. Finally, there can be a number of `abort/restart` actions due to the need to free the system from transactions that cannot meet the deadlines; therefore, some kind of resource preclaiming can be used.

Concluding this short introduction to RTDBMS, we must note that there is a strong interaction among system functions such as resource scheduling, buffer management, and concurrency control; this forces the designer to simulate the combined adoption of different algorithms under variable workloads in order to produce an efficient and effective system.

2.3.4 Main Memory Databases

In Sect. 2.3.3, we mentioned that latency in disk accesses can be an obstacle to meeting deadlines in RTDBMS, but another reason for avoiding disks in pervasive



Disk resident DB access

Main memory resident DB access

Fig. 2.9 How data are accessed in a Disk Based Data Base (DBDB) and in an MMDB

systems is that often their size is not compatible with the size of the other devices in the system (e.g., sensors, wearable or implantable systems). Therefore, a main memory (MM) database becomes a valid alternative for permanent data storage and retrieval. In an MM database (MMDB), data are stored permanently in main memory using specialized data structures that are totally different from those of data cached in main memory by a disk-resident DBMS and that follow a different access strategy. Figure 2.9 shows the two access modalities: cashed data are accessed through indexes designed for disk access; a buffer manager checks if the relevant data block is in the cache; if not, an I/O operation is started, and if yes, data are directly transferred to the application working area. In MMDB, data are accessed by directly referring to their memory address [1, 9].

2.3.4.1 Structure of MM Relational Tables

Relational tables are usually stored as flat files by the file system; this means that the values of the attributes are embedded in the tuple and tuples are stored sequentially in the file. The result is that the file system must manage sequential variable length record files. Moreover, many of the stored values are replicated many times with an obvious waste of storage space. If this cannot be a serious drawback on disks, it becomes annoying with MMDBs, where storage space is still precious, but, on the contrary, the access time to information is uniform.

Domain tables have been defined [4, 14] in order to overcome these problems; a domain table stores all the values taken by a given domain only once, while the original relational tables only store pointers to the values in the domain tables (Fig. 2.10). With such an organization, we get two benefits: (1) value redundancy is eliminated and (2) the relational tables only store fixed length pointers for all the enumerated types larger than the pointer size, resulting in fixed length tuples, which can be easily managed. Moreover, the domain tables can be shared among different

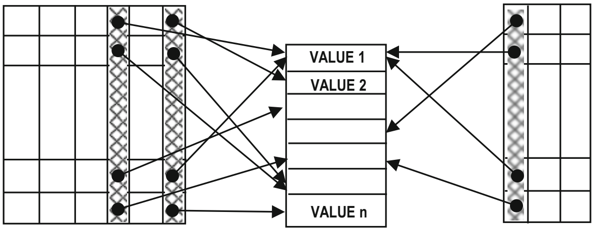


Fig. 2.10 Domain table storage

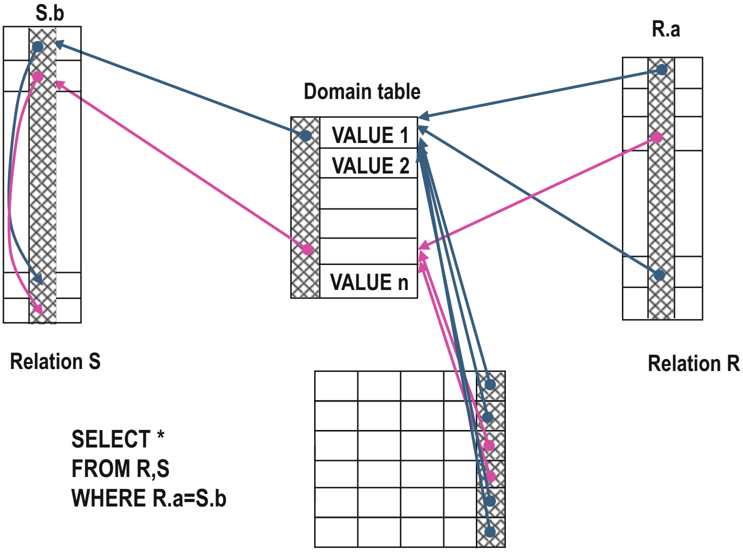


Fig. 2.11 Nested-loop join with ring indexing

columns of the same table or even among columns belonging to other relational tables, resulting in even more memory savings.

The usage of domain tables and pointers has also a side effect on the execution of relational operators. Figure 2.11 shows how a domain table together with pointers linking the same values inside a relational table creating a ring index makes the execution of a nested-loop join operation very fast. This is a very different situation with respect to disk-resident databases where nested-loop joins are very expensive in terms of I/O accesses.

Besides hashing, several variants of the T-tree [15] indexing structure have been proposed for MMDBMS. The common goal is that of keeping the tree balanced, in order to obtain a fast access to its elements, while minimizing the frequency of rebalancing operations after the insertion or deletion of data items.

Last, but not least, we want only to mention that the advent of flash memory technologies brought new burdens to file systems. Unlike EEPROM-RAM technologies, flash memories are nonvolatile, but they are asymmetric in reading and writing; writing can only be made in blocks, and blocks cannot be modified but must be erased and written as new. Moreover, current flash units have a lifetime between 10^5 and 10^6 write/erasure cycles, and this forces the file system to adopt wear leveling policies, dynamically remapping blocks in order to spread write operations between sectors to avoid localized block failures [8].

2.4 Data Analytics

Once the structural meaning of the pervasive data is understood, another important aspect of pervasive data management is the actual extraction of synthetic knowledge through *massive data analysis and processing*. Indeed, there is often information "hidden" in the data that is not readily evident, and discovering it manually may take weeks or months.

Data analysis is the process of inspecting, cleaning, transforming, and modeling data with the goal of highlighting useful information, suggesting conclusions, and supporting decision making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains. The main methods of data analysis are *data mining* [7, 30], *exploratory data analysis* [31], and *business intelligence* [12, 24].

2.4.1 Data Mining

Data mining focuses on modeling and knowledge discovery by studying algorithms and techniques to find interesting patterns that represent implicit knowledge stored in massive data repositories, useful to generate concise descriptions (*models and patterns*) of the analyzed data. Data mining draws ideas from *machine learning, pattern recognition, databases, and statistics*. The purposes of data mining are *descriptive* and *predictive*. The former aims at summarizing the data into smaller, useful pieces of information, while the latter studies recent and historical data, allowing predictions about the future. The main data mining techniques are:

- *Classification* is a predictive method. Given a collection of records and their attributes, the idea is that the system is trained to guess the value of a specific attribute (its *class)* given the values of the other ones. An example from the pervasive systems domain is for the system to guess the category of a user (expert or nonexpert) based on the data of their previous queries on the art pieces.
- *Clustering* is a descriptive technique which, given a set of data items with a set of attributes and a similarity measure among them, groups the data items into *data clusters* such that the data items in the same cluster are similar to one another according to the similarity measure, while any two items that are in different

clusters are far from each other according to the same measure. An example of the use of this technique is distributing documents describing art pieces into different groups according to how many terms they have in common. The documents of one such group will probably refer to the same topic.

- *Association-rule discovery* is a descriptive method by means of which, given a set of records each of which contains a number of items from a given collection, we can extract *dependency rules* that highlight interesting relations between the items of the same record. For example, the rule $\{Warhol, Chagall\} \Rightarrow \{VanGogh\}$ found in the exhibition reservation data would indicate that if a visitor reserves tickets for Warhol and Chagall exhibitions together, she/he is likely to also reserve a ticket for a Van Gogh exhibition.
- *Sequential pattern discovery* is also a descriptive method, similar to association-rule discovery but with the addition of the time dimension. In this case, the rules are formed by discovering patterns of events that are related by timing constraints. For example, the sequential pattern $\{exhibition, enoteca\}$—`6hours`—$\{concert\}$ indicates that many people like going to a concert about 6 h after having visited an exhibition and having tasted typical local wine at an enoteca.
- *Regression* is a predictive technique directly taken from statistics, by means of which the value of a variable is described as a function of another set of variables. For instance, the weather of the next day can be described as a function of wind speed, temperature, humidity, and air pressure.
- *Anomaly detection* is also a predictive method, where significant deviations from normal behavior are considered as symptoms of some anomaly that should be further investigated. For example, movements in a museum room, sensed at a time in which there are no visits, might indicate some mischief.

2.4.2 Other Common Data Analysis Methods

Business intelligence relies heavily on aggregation: online analytical processing (OLAP)[23] is typically performed over a *data warehouse*. A data warehouse is a single, complete, historical, and consistent store of data obtained from a variety of different sources to the end of decision support and is normally maintained separately from the organization's operational database. The presence of historical data, i.e., not only the current value of each datum but also its previous versions, is the key to understanding business trends over time.

Data exploration and exploratory data analysis suggest a preliminary exploration of the data to better understand its characteristics. Introduced by John W. Tukey, it is a field of statistics where the role of the researcher is to explore the

[23] OLAP queries are typically aimed at sophisticated analyses and mathematical computations over different dimensions and hierarchies of the data.

data in many possible ways—including the use of graphical visualizations like histograms and diagrams or pictorial tools like colored maps to illustrate the spread of a phenomenon (e.g., population wealth) in the various areas of the world.

The abstractions and languages that are currently proposed to manage large collections of data are heavily influenced by the underlying ICT platforms, but they are unsuitable for supporting computational interdisciplinarity, that is, the ability to use the best of, e.g., analytical, inductive, and simulation techniques, all at work on the same data. Most of the proposed analysis methods and techniques are based on two main assumptions: data are structured, and the way they are structured is closely related to the kind of analysis that should be carried out. Clearly, these two assumptions are not valid in the pervasive environment, where data are integrated from heterogeneous and unstructured sources and may not be generated by well-defined processes, as in general they are used to satisfy unanticipated and different requirements. In such a scenario, research is needed to make the existing analysis techniques more flexible.

2.4.3 Making Sense of Big Data

Results of queries on pervasive data can be overwhelming, especially when they are issued online by users who want effective answers on small-size displays. Actually, the term *information overload* was already used by Alvin Toffler in his book *Future Shock*, back in 1970, and refers to the difficulty in understanding and making decisions when too much information is available: When the amount of data is too large, even the interpretation of the answer to a query may be nontrivial, since the returned dataset is simply too big to be easily readable. As a consequence, data analysis techniques have also been put at the service of the nonexpert, by means of simple interfaces that support various kinds of nonstandard query answers, among which we can quote *faceted search*, also called faceted navigation [32]: a strategy of exploring sets of information items based on a taxonomy, i.e., a classification system. Items are classified according to their “facets,” i.e. (possibly hierarchical) attributes: in the case of cultural heritage items, these could be “painting school” or “subject matter.” The user can browse the set making use of the facets’ “values” (e.g., “expressionism” and “portraits”) as the selection criteria, and the system provides various kinds of visual and interaction tools to give her/him an idea of what the selected set of items is about.

Another approach aimed at easing final users in making sense of the data is that of *intensional answers* [22] and later extended to *approximate intensional answers* [19]. With this method, *intensional information*[24] is produced from an original

[24] The term *intension* suggests the idea of denoting objects by means of their properties rather than by exhibiting them. Thus, the intensional characterization replaces a lengthy list of items with a succinct description.

dataset by using data mining technique to collect the properties that hold for the majority of the data, and then the gained knowledge is used to answer the query in a synthetic way. For example, the short answer to a user who wants to know about the contents of the museums of Sicily might be that "80% of the items present in the museums of Sicily are from the Greek period."

A way to *reduce* the amount of data delivered to the user, instead of synthesizing them, is context-aware data tailoring, where, given a target application, in each specific context, the system provides the user only with the set of data (sensor readings, environmental information, available services, close-by people, points of interest, etc.), which is relevant for that application in that context. Moreover, the relative importance of information to the same user in different contexts or, reciprocally, to different users in the same context may vary enormously; thus, another way to ease the information overload problem is *information personalization*, i.e., data reduction, ranking, and reshaping according to users' (possibly contextual) recommendations and preferences. This topic will be dealt with in Part IV of this book.

References

1. Special issue on main-memory database systems. Data Eng. **36**(2), 6–62 (2013)
2. Brewer, E.: Pushing the cap: strategies for consistency and availability. Computer **45**(2), 23–29 (2012)
3. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: Proceedings of Symposium on Operating Systems Principles, SOSP, pp. 205–220 (2007)
4. DeWitt, D.J., Katz, R.H., Olken, F., Shapiro, L.D., Stonebraker, M., Wood, D.A.: Implementation techniques for main memory database systems. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 1–8. ACM, New York (1984)
5. Doerr, M., Gradmann, S., Hennicke, S., Isaac, A., Meghini, C., van de Sompel, H.: The europeana data model (edm). In: World Library and Information Congress: 76th IFLA General Conference and Assembly, pp. 10–15 (2010)
6. Doreian, P., Stokman, F.: Evolution of Social Networks. Routledge, London (2013)
7. Freytag, J.C., Ramakrishnan, R., Agrawal, R.: Data mining: the next generation. Inf. Technol. **47**(5), 308–312 (2005)
8. Gal, E., Toledo, S.: Algorithms and data structures for flash memories. ACM Comput. Surv. **37**(2) (2005)
9. Garcia-Molina, H., Salem, K.: Main memory database systems: an overview. IEEE Trans. Knowl. Data Eng. **4**(6), 509–516 (1992)
10. Hadim, S., Mohamed, N.: Middleware: middleware challenges and approaches for wireless sensor networks. IEEE Distrib. Syst. Online **7**(3), 1 (2006)
11. Han, J., Haihong, E., Le, G., Du, J.: Survey on nosql database. In: Proceedings of the 6th IEEE International Conference on Pervasive Computing and Applications (ICPCA), Port Elizabeth, pp. 363–366, 26–28 October 2011
12. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. Wiley, New York (1996)
13. Kourouthanassis, P.E., Giaglis, G.M.: A design theory for pervasive information systems. In: Proceedings of International Workshop on Ubiquitous Computing (ICEIS), pp. 62–70 (2006)

14. Lehman, T.J., Carey, M.J.: Query processing in main memory database management systems. In: Proceedings of 1986 ACM SIGMOD International Conference on Management of Data, pp. 239–250 (1986)
15. Lehman, T.J., Carey, M.J.: A study of index structures for main memory database management systems. In: Proceedings of the Twelfth International Conference on Very Large Data Base, Kyoto, pp. 294–303 (1986)
16. Lindström, J.: Relaxed correctness for firm real-time databases. In: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing, pp. 82–86 (1996)
17. Lindström, J.: Real Time Database Systems, pp. 1–36. Solid-IBM, Helsinki. www.cs.helsinki.fi/u/jplindst/papers/rtds.pdf (2008)
18. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1), 122–173 (2005)
19. Mazuran, M., Quintarelli, E., Tanca, L.: Data mining for XML query-answering support. IEEE Trans. Knowl. Data Eng. **24**(8), 1393–1407 (2012)
20. McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D., Barton, D.: Big data. The management revolution. Harv. Bus. Rev. **90**(10), 61–67 (2012)
21. Moniruzzaman, A., Hossain, S.A.: Nosql database: new era of databases for big data analytics-classification, characteristics and comparison. arXiv preprint (2013) [arXiv:1307.0191]
22. Pirotte, A., Roelants, D., Zimányi, E.: Controlled generation of intensional answers. IEEE Trans. Knowl. Data Eng. **3**(2), 221–236 (1991)
23. Ramamritham, K., Son, S.H., DiPippo, L.C.: Real-time databases and data services. Real-Time Sys. **28**(2–3), 179–215 (2004)
24. Rizzi, S.: Data warehouse. In: Wiley Encyclopedia of Computer Science and Engineering. Wiley, New York (2008)
25. Rosi, A., Mamei, M., Zambonelli, F., Dobson, S., Stevenson, G., Ye, J.: Social sensors and pervasive services: approaches and perspectives. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Seattle, pp. 525–530, 21–25 March 2011
26. Schreiber, F.A., Camplani, R., Fortunato, M., Marelli, M., Rota, G.: Perla: a language and middleware architecture for data management and integration in pervasive information systems. IEEE Trans. Softw. Eng. **38**(2), 478–496 (2012)
27. Schroth, C., Janner, T.: Web 2.0 and soa: converging concepts enabling the internet of services. IT Prof. **9**(3), 36–41 (2007)
28. Shen, C.C., Srisathapornphat, C., Jaikaeo, C.: Sensor information networking architecture and applications. IEEE Pers. Commun. **8**(4), 52–59 (2001)
29. Stankovic, J.A., Son, S.H., Hansson, J.: Misconceptions about real-time databases. Computer **32**(6), 29–36 (1999)
30. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Boston (2005)
31. Tukey, J.W., Wilk, M.B.: Data analysis and statistics: an expository overview. In: American Federation of Information Processing Societies: Proceedings of the AFIPS '66 Fall Joint Computer Conference, 7–10 November 1966, San Francisco, pp. 695–709 (1966)
32. Tunkelang, D.: Faceted search. In: Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, San Rafael (2009)
33. Ulusoy, O.: An annotated bibliography on real-time database systems. SIGMOD Rec. **24**(4), 40–45 (1995)
34. Xiong, M., Sivasankaran, R.M., Stankovic, J.A., Ramamritham, K., Towsley, D.F.: Scheduling transactions with temporal constraints: exploiting data semantics. In: Proceedings of the Real-Time Systems Symposium, RTSS, pp. 240–251 (1996)

Chapter 3
Privacy in Pervasive Systems: Social and Legal Aspects and Technical Solutions

Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, Stefano Paraboschi, and Pierangela Samarati

3.1 Introduction

In today's society, most actions we perform are recorded and the collected data are stored, processed, and possibly shared in a way that was impossible until a few years ago before the development of ubiquitous and pervasive technologies. Ubiquitous technologies represent one of the most significant revolutions in information and communication technologies. The term "ubiquitous computing" was introduced by Mark Weiser in the late 1980s to describe a future world based on "the idea of spreading computers ubiquitously, but invisibly, throughout the environment" [48]. Pervasive and ubiquitous technologies are now present everywhere in our daily life. People may have several devices (e.g., smartphones, tablets) that can be used to access any kind of services anywhere, anytime. There are also devices that can keep track and measure health conditions (e.g., blood pressure and heart rate) and send such information to different parties. The amount of data that is therefore generated everyday has grown exponentially and is expected to continue in the coming years. Since the cost of data storage and processing has significantly decreased, all these data can be stored long term and made accessible when needed.

While the technology advancements and the possibility of collecting, storing, processing, and accessing data everywhere in the world bring enormous benefits, users are becoming more and more concerned about their privacy. In fact, collected data can be used to identify individuals or infer something that was not intended

S.D.C. di Vimercati (✉) • S. Foresti • G. Livraga • P. Samarati
Università degli Studi di Milano, Via Bramante 65, 26013 Crema, Italy
e-mail: sabrina.decapitanidivimercati@unimi.it; sara.foresti@unimi.it; giovanni.livraga@unimi.it; pierangela.samarati@unimi.it

S. Paraboschi
Università degli Studi di Bergamo, Via Marconi 5, 24044 Dalmine, Italy
e-mail: parabosc@unibg.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_3

for disclosure. The location information generated by a cell phone, the pattern of walking as recorded by a surveillance camera, and the combination of seemingly innocuous information (e.g., the ZIP code and the date of birth) are all examples of data that can be exploited to identify the person to whom they refer, not to mention information like the biometrics that may raise some privacy concerns (e.g., [5, 20, 24, 42]). The main motivation behind these privacy issues is that when users, for example, subscribe to a new social networking service or provide some information to access a service, they immediately lose control over their data.

The users' abilities to manage their personal information and also to delete such information may then become difficult, if not impossible. This is a well-recognized problem that research and development communities, governments, and public and private organizations are all trying to solve. In particular, at the European level, a proposal for a regulation was released in 2012 with the aim of unifying all the data protection laws within the European Union with a single General Data Protection Regulation (http://ec.europa.eu/justice/data-protection/). The main goal of this new regulation on data protection is to take into consideration the recent technological developments (e.g., cloud computing and social networks) and their security and privacy risks [21, 30] to build trust in the digital world and to empower users to keep control over their data. There are several key aspects that are considered in the proposed regulation such as the introduction of new concepts (e.g., encrypted data and genetic data), which address new privacy concerns; the right to be forgotten, which allows users to require the erasure of their personal data whenever, for example, such data are no more necessary for the purpose for which they have been collected; and the applicability of such a regulation also to companies based outside the EU that process the personal data of EU residents.

Clearly, privacy in the modern digital society is a complex concept that should be addressed from several points of view: legal, social, economical, and technological. The main focus of this chapter is on the technological aspect of privacy within today's ubiquitous and pervasive systems. In particular, we aim to analyze the main privacy issues that can arise when collecting, processing, and sharing data in pervasive and ubiquitous environments and then at presenting available technological solutions that can be put in place to counteract them. As a running example, we will consider a museum that manages a large cultural heritage. The museum aims to exploit the pervasive availability of computing infrastructures to develop a framework for providing a cultural site (e.g., indoor museums, archaeological sites, historical archives, old town centers) with several *smart* services for assisting users (e.g., visitors or staff personnel) in the seamless exploration and management of the related environment. In this context, smart and pervasive solutions can be adopted, for example, from the electronic management of ticket purchases, to interactive and guided tours based on the sensed proximity of a visitor to a specific exhibition, to the continuous monitoring of environmental conditions (e.g., humidity, temperature, pollutant concentrations) in the museum premises through sensors.

The remainder of the chapter is organized as follows: Section 3.2 illustrates our reference scenario and presents the main related privacy risks. Section 3.3 provides an overview to the most well-known approaches to protect location information.

Section 3.4 discusses some solutions that allow the privacy-preserving sharing of personal/sensitive data. Section 3.5 shows possible approaches that allow the secure storage of personal/sensitive data. Finally, in Sect. 3.6 we provide our final remarks and conclude the chapter.

3.2 Privacy in Pervasive Systems

We first introduce the reference scenario that will be considered in the remainder of this chapter (Sect. 3.2.1). We then illustrate the privacy issues that may arise in such a scenario (Sect. 3.2.2).

3.2.1 Reference Scenario

Our scenario (Fig. 3.1) refers to a museum with both indoor and outdoor exhibitions and facilities, distributed in a wide geographical area, such as a city or a region (e.g., Rome, Paris, New York). To be considered "smart," the museum features a digital infrastructure providing digital services and pervasive solutions to both enhance the experience of the visitors and efficiently manage the museum and its exhibitions. As illustrated in Fig. 3.1, the considered scenario is characterized by the interaction of different subjects. In particular, users (i.e., visitors to the museum) interact with the museum system through an app that they can install on their smartphone and by enabling GPS and location services. The app acts as a smart guide, as described

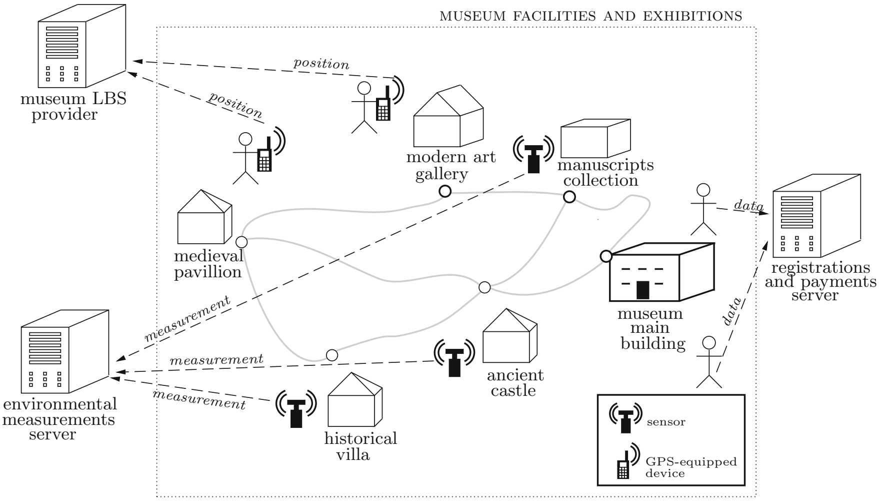


Fig. 3.1 Reference scenario

in the following. Also, the museum features a set of environmental monitoring stations to measure different environmental parameters (e.g., temperature, humidity, pollutants). Data about users and the environmental measures collected by the museum are stored at different servers (i.e., the registrations and payments server, the location-based service (LBS) provider, and the environmental measures server). The smart solutions adopted by the museum can be classified in the following three groups, depending on their objective:

- *Ticket purchases and visitor registrations.* Visitors to the museum can buy their tickets either on spot or online. When buying online, users pay by credit card and can then collect their tickets presenting the credit card used for the payment at an automatic machine at any of the museum facilities. When purchasing a ticket, a visitor becomes a user of the museum systems. Users can also register to the museum web site and/or follow it on social networks to receive news, discounts, and other information. Data about users and registrations are stored at the registrations and payments server, which is in charge of maintaining all information provided by users. Figure 3.1 illustrates the information flows caused by ticket purchases and registration activities by users as dashed arrows, labeled *data*, from users to the registrations and payments server. Figure 3.2 illustrates an example of the relation stored and managed by this server. The relation stores personal data (attributes `Phone`, `Name`, `DoB`, and `Sex`), ticket information (attribute `TicketType`), and payment information (attribute `Payment`) about the visitors to the museum. Attribute `TicketType` represents the kind of ticket bought by a visitor, which can be either *regular* (i.e., full price and no discount) or discounted/free if, for example, the visitor suffers from specific pathologies (*health* ticket) or has a particular job (e.g., *army* or *government*) for which the museum adopts reduced fares.
- *Smart guides to artworks.* Besides traditional guided tours to the different exhibitions, in which an authorized guide escorts the visitors, the museum also offers location-based and automatic guided tours. To this end, visitors can download an ad hoc location-based app, provided by the museum, on their smartphone (*smartphone* icon close to users in Fig. 3.1), which describes the

Phone	Name	DoB	Sex	ZIP	TicketType	Payment
(800) 917-5551	Alice	1960/04/10	F	97401	Health	Credit card
(500) 234-5678	Bob	1970/05/12	M	98302	Army	Debit card
(541) 271-2136	Carol	1960/04/04	F	97467	Regular	Cash
(360) 474-4614	Daniel	1970/05/20	M	98245	Army	Cash
(360) 373-2030	Erik	1970/07/12	M	98312	Navy	Cash
(541) 946-1711	Fred	1960/04/11	F	97434	Professor	Credit card
(360) 435-3746	Greg	1970/07/25	M	98223	Government	Check
(253) 863-5555	Hal	1970/07/30	M	98389	Marines	Cash
(360) 794-7058	Ian	1970/05/12	M	98290	Army	Credit card
(503) 497-91 33	John	1950/12/01	M	97210	Air Force	Debit card

Fig. 3.2 Sects. 3.3 (a), 3.4 (b), 3.5 (c)

artworks based on the position of the user. This indeed represents a great advantage for the visitors to the museum, who can decide their own itineraries without the need of reserving a guide in advance while, at the same time, enjoying professional illustrations of the exhibitions based on their position. Visitors can then walk around in the city, and as soon as they approach an artwork (such as the Trevi Fountain, the Sistine Chapel, or the Colosseum in Rome), their location-based app will ring an alarm and a description of the artwork will start. The LBS offered by the museum can also suggest to visitors the best itinerary to avoid queues that would delay their visit. To this end, it collects and aggregates location data about the users who are using the location-based app and takes them into consideration when determining the best itinerary to be suggested to a new visitor. For instance, it can rearrange the itinerary of visitors (e.g., if a place is too crowded, an updated itinerary skipping that place can be suggested to a visitor). Figure 3.1 illustrates the communications from users to the LBS provider of the museum to support the smart guide service as dashed arrows, labeled *position*.

- *Environmental monitoring*. To protect the artworks, the museum uses a pervasive environmental monitoring system (*sensor* icons distributed in the environment in Fig. 3.1). This system analyzes and keeps under control different parameters that might harm the artworks such as the temperature of a room to regulate air conditioning, the quality of the air (e.g., specific pollutants and humidity to enforce specific countermeasures to protect the artworks), and the number of visitors at an exhibition or in a given room to regulate further access to the same. Figure 3.1 illustrates the communication exchanges from sensors to the environmental measurements server as dashed arrows, labeled *measurement*. Figure 3.3 illustrates an example of the relation kept by this server storing the measurements of temperature (in Celsius degrees), humidity percentage, concentration of PM10 pollutant (in μg/m^3), and noise pollution (in dB).

The data about users and environment are *analyzed* (either at runtime, such as for the environmental sensing, or offline, such as for discovering statistics on the visitors based on the contact information provided at the time of purchasing tickets), *stored*, and *maintained* for possible future use, possibly including disclosure to third parties. For instance, the ministry of arts and culture periodically asks the museum

Fig. 3.3 Environmental measurements at the facilities of the museum

Sensor	**Temp** (^oC)	**Humidity**	**PM10** ($\mu g/m^3$)	**Noise** (dB)
int_A	25	40%	25	60
ext_B	28	60%	30	55
int_C	27	45%	40	57
ext_D	30	55%	50	62
ext_E	29	53%	55	58
int_F	22	42%	59	32
ext_G	30	59%	50	47
ext_H	28	60%	42	50
int_I	28	43%	58	30
int_J	22	51%	35	35
ext_K	32	63%	37	65

to provide all data related to visits and payments. Thanks to these data, the ministry can study marketing strategies and take knowledge-based decisions regarding, for example, special rates and discounts for specific groups of visitors, increasing the personnel working in the museum, adjusting the opening hours, planning special exhibitions, and many other activities. The museum can also decide to *share* these data with third parties. For instance, the museum can provide its data to research organizations to study countermeasures for improving the quality of the air by reducing the concentration of specific pollutants. The paths of users among the different facilities of the museum can be shared with other museums to suggest to each user the most appropriate smart visit based on their previous ones.

3.2.2 Privacy Issues

The main privacy issues that arise in the considered scenario are related to the fact that the data collected by the museum include sensitive information that can put at risk the privacy of users to whom it refers. The collected data span from the *contextual information* generated by the pervasive infrastructure to the information released by the users themselves. Contextual information is needed to develop smart services that can react to the environment surrounding a user. A notable example of this kind of information is the *location information* that users continuously release during their visits. Such information can then be used to track the movements of users, which is considered intrusive and harmful to their privacy. The data released by the users are needed to take advantage of the museum services. The storage and processing of these data should always be performed in respect of the privacy of the users. For instance, information like phone numbers cannot be shared with an advertising company without the prior consent of the users. When accessing sensitive information, both *direct* and *indirect* privacy violations may occur, as illustrated in the following:

- *Direct violations.* Direct violations are caused by the presence in the collected data of sensitive information available to all parties accessing the data. For instance, all recipients accessing the data collection in Fig. 3.2 can discover the phone numbers of the visitors to the museum.
- *Indirect violations.* Indirect violations are caused by the possibility of determining sensitive information that is not explicitly included in the collected data but can be obtained from them. For instance, by observing the discounts applied to the tickets purchased by visitors to the museum in Fig. 3.2, a recipient can infer that *Alice* suffers from a disease and that *Bob* is from the military.

It is interesting to note that privacy violations might affect both individuals represented in the collected data (i.e., registered visitors) and individuals that are apparently not involved in the data release. For instance, the relation in Fig. 3.2 includes personal and payment information of the visitors to the museum, and its improper sharing or distribution can affect the privacy of the visitors. As another example, consider the table in Fig. 3.3. An insurance company might increase the premium to individuals living in the areas close to the museum and for which the table reports a high PM10 concentration. This behavior clearly affects the privacy of individuals who are not necessarily visitors to the museum.

As might be clear from the discussion above, privacy violations can occur for a variety of different reasons, including the presence of sensitive information in the data collection (e.g., attributes `Phone` and `TicketType` in the relation in Fig. 3.2), the existence of *correlations* and *associations* among different datasets (e.g., correlations among pollutant concentrations and respiratory diseases), and the observation of data *evolution*. As an example of this latter aspect, suppose that environmental sensor *ext_B* of the museum (see Fig. 3.3) be close to the city railway and record noise levels continuously at regular time intervals. Assume also that the schedule of freight trains be sensitive and therefore not publicly available. By observing peaks in the sensed noise levels and linking them to the (public) timetables of passenger trains, it might be possible to deduce the schedule of freight trains. *Unusual* data can also leak sensitive information: for example, an individual paying the museum ticket with a very exclusive credit card makes her/him stand apart from others and reveals that, in all probability, she/he enjoys a relatively high income.

In the remainder of this chapter, we survey some of the approaches that can be adopted to protect data and users from the privacy issues described above. To guide the reader through the chapter, Fig. 3.4 illustrates a summary of the solutions that will be described in the remaining sections.

3.3 Protecting Location Information

The widespread adoption of mobile communication devices and the advancements made on location technologies have contributed to the development of a great variety of LBSs for business, social, or informational purposes. As an effect of such innovative services, however, privacy concerns are increasing. In fact, location information is subject to a variety of privacy threats, including stalking or physical harassment. Location information can also be exploited for inferring sensitive information about users. As an example, consider a user running the location-based app provided by the museum for smart guides in Rome. Since the exhibitions and facilities of the museum are distributed over the city, while walking from the Colosseum to Trevi Fountain, the user might stop by a pharmacy selling medicaments for a specific disease, thereby releasing her/his position to the LBS of the museum. While this might not be a problem when the service provider (the museum, in our example) is trusted by the user, this situation becomes problematic when location information is shared with (or managed by) third parties. Anyone accessing such location information can in fact infer that the user (or an individual close to her/him) suffers from that specific disease. The existing solutions for protecting privacy of location information can be classified based on whether they

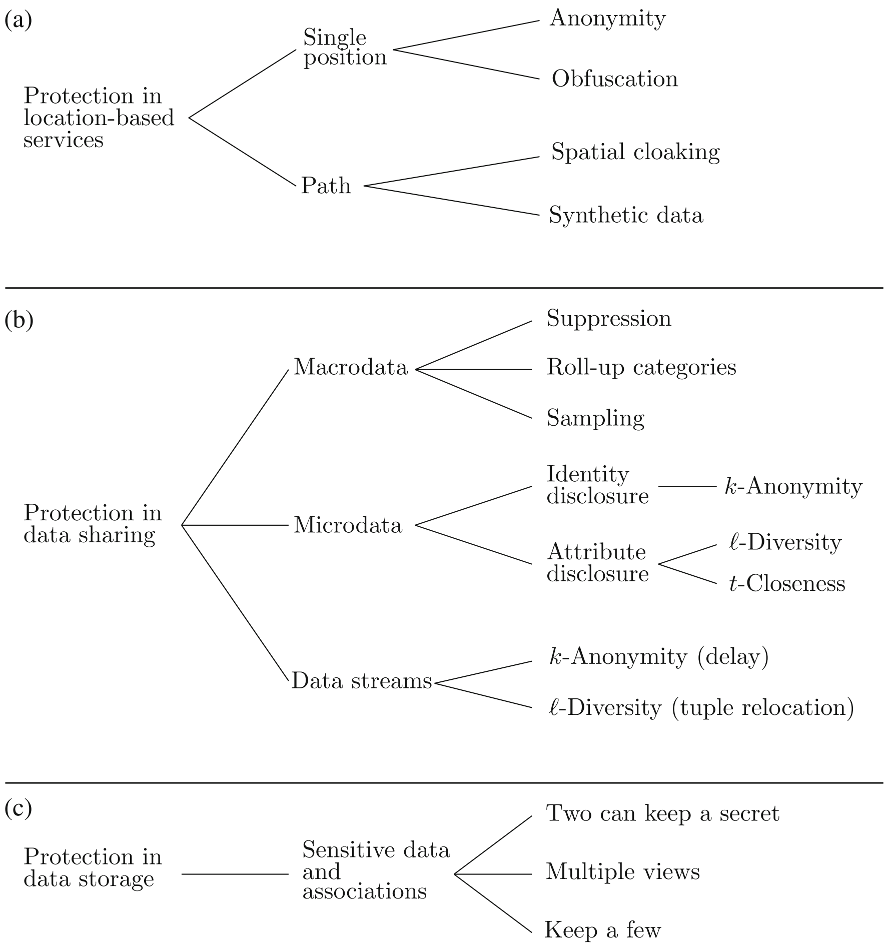


Fig. 3.4 Summary of the solutions illustrated in Sects. 3.3–3.5 (**a**)–(**c**)

aim at protecting the single positions of a user or her/his path whenever she/he continuously releases the trace of her/his movements to a provider. In our running example, the first class of solutions are important if the user decides to use the smart guide app in pull mode, that is, the app issues a query to the LBS of the museum when the user is close to an artwork. The second class of solutions can instead be useful when the user decides to use the smart guide in push mode. To this end, the app continuously sends the user location to the LBS of the museum, and as soon as she/he reaches a point of interest, the app automatically receives the description for that art piece.

We will now illustrate the most well-known approaches for protecting location privacy, distinguishing between solutions tailored to protect the single position of a user and those aimed at protecting her/his path.

Single Position Protection *Anonymity-based* techniques (e.g., [3, 6, 8, 25, 28, 33, 39, 40]) aim at protecting the association between users' identities and their precise position to prevent reidentification by observing users' requests to the LBS. These techniques include solutions based on the concept of k-anonymity [13, 43] originally proposed in the database context (see Sect. 3.4.2). To protect users' identities, their explicit identifier is removed and the precision of their position is degraded in such a way that a user is indistinguishable by other $k - 1$ users in a given location area or temporal interval. Figure 3.5a illustrates an example of the application of such protection techniques, where $k = 4$. In the figure, users are represented by a small circle, labeled with the user name. On the right-hand side of the figure, users *Greg*, *Hal*, *Ian*, and *John* are de-identified (i.e., their identities are not associated with their queries), and all queries are associated with the area represented by the gray rectangle in the figure. Therefore, every request can be indistinguishably

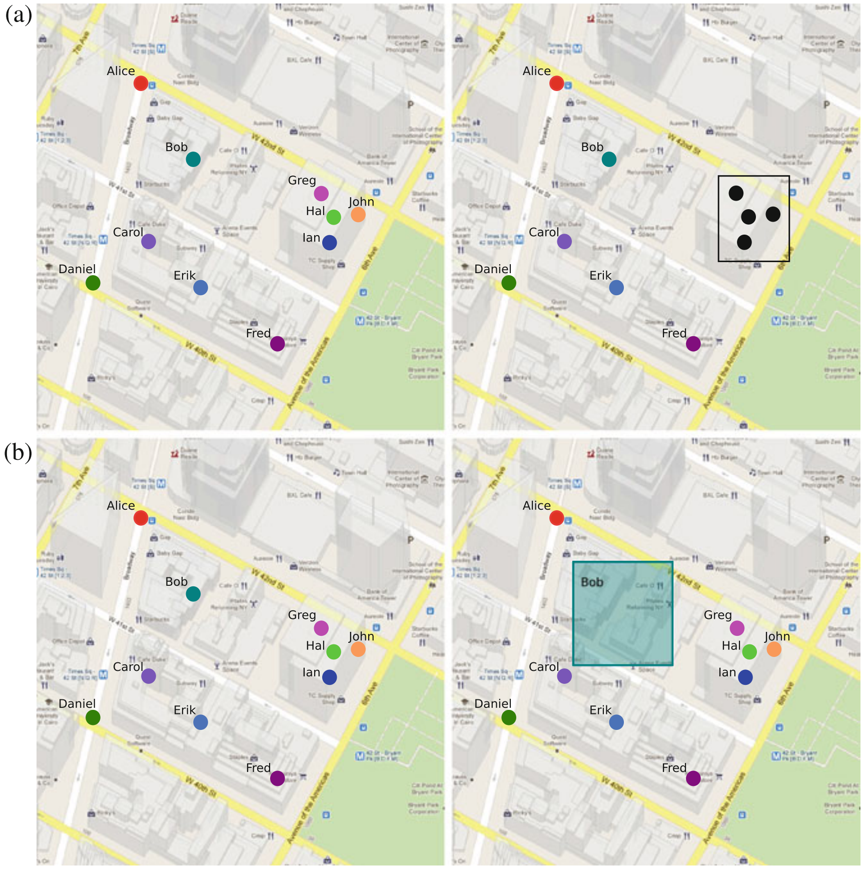


Fig. 3.5 Protecting users' location through anonymity-based (**a**) and obfuscation-based (**b**) approaches

generated by any of the four users. Whenever the identity of a user needs to remain attached to her/his location information (e.g., with reference to our scenario, when descriptions of exclusive art pieces should be available only to specific visitors who paid an additional ticket), *obfuscation-based* techniques (e.g., [2, 4, 22]) can be adopted instead of anonymity-based solutions. Rather than anonymizing users, these techniques degrade the accuracy of their location. The main goal of these techniques is therefore to perturb the location information of the users while still maintaining a binding with their identity. Figure 3.5b illustrates an example of the application of these techniques. The right-hand side of the figure shows a degradation of the released position of user *Bob*, represented by the shaded rectangle, so as to protect his actual position. Note that, as opposed to Fig. 3.5a, the identity of *Bob* is not hidden to the LBS provider and remains associated with his (degraded) position.

Path Protection The protection of the trajectory information of a user is a critical aspect in our reference scenario. Suppose that the users adopt the museum app in push mode, meaning that the app on their smartphone continuously sends their positions to the LBS offered by the museum. While walking around the city and sightseeing, a user might visit other places that can be considered sensitive as they can be exploited to infer personal information about her/him. For instance, the user can stop to a pharmacy selling drugs for rare diseases, hence making this information available to all parties observing her/his movements. In this scenario, it is possible to adopt path-protecting approaches. Figure 3.6 illustrates an example of the application of such protection techniques, in which the real paths followed by *Alice* and *Bob* on the left-hand side of the figure have been protected by appearing indistinguishable to the eyes of an observer. Traditionally, these solutions use *spatial cloaking* techniques: a cloaked spatial region must be shared by at least *k* users, and to protect user trajectories, all *k* users must appear as belonging to the same region as time passes (e.g., [11, 41, 49]). A different approach is based instead on the

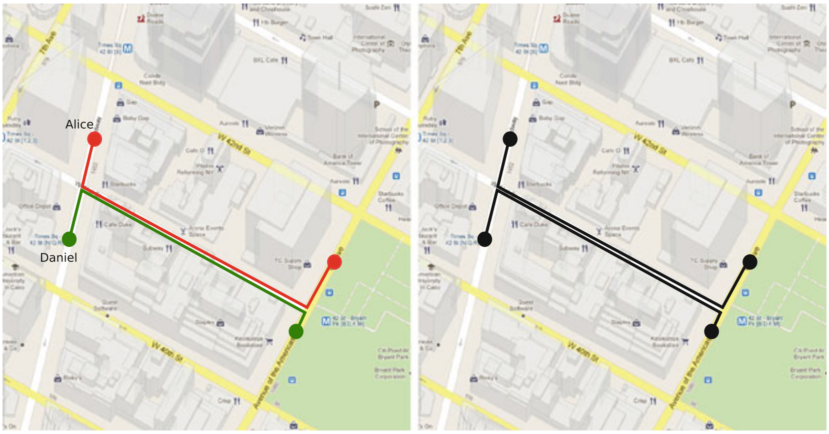


Fig. 3.6 Protecting users' path

generation, and release to the LBS, of (partially) *synthetic* trajectories. For instance, the technique in [38] relies on mix zones created over synthetic trajectories, obtained with first-order Markov chains from historical data. The release of fake paths is also at the basis of a recent technique aimed at counteracting the risk of sensitive information disclosure due to the observation of unusual paths. In fact, being unusual with respect to what is expected and considered common, these paths can leak information not intended for disclosure. The proposal in [5] introduces a framework, based on first-order Markov chains, to evaluate how "unusual" a path followed by a user is with respect to traditional trajectories (in our example, common itineraries followed by visitors) to reduce the risk of inferences by releasing a slightly modified and safe (i.e., less unusual) path.

3.4 Privacy-Preserving Data Sharing

Data sharing and dissemination are becoming more and more common and, in some cases, even mandatory by law. Collected data can be disseminated in the form of *macrodata* or *microdata* [14]. Macrodata are *aggregate* values representing statistics of interests computed over a sample population. Such statistics are measures that summarize the values of one or more properties/attributes of *respondents* (i.e., individuals, organizations, associations, business establishments, and so on). Microdata are specific data related to single respondents (i.e., single visitors, in our example). The release of macrodata and/or microdata might cause leakage of sensitive information that was not intended for disclosure. In this section, we will illustrate available solutions for protecting macrodata (Sect. 3.4.1) and microdata (Sect. 3.4.2) and for protecting data streams, which are common in pervasive scenarios since data are often collected by sensing devices in streams (Sect. 3.4.3).

3.4.1 Protecting Macrodata

Macrodata are represented as tables where each cell of a table is the value of a quantity computed over the considered properties. A macrodata table usually includes marginal totals, that is, the aggregate computed over each row/column in the table. Depending on how macrodata tables are defined, they can be classified as (1) *count* and *frequency tables*, where each cell contains the *number* (*percentage*, respectively) of respondents that share the same value over all attributes of analysis reported in the table, and (2) *magnitude tables*, where each cell contains an *aggregate value* (e.g., sum) of a *quantity of interest* over all attributes of analysis reported in the table. Figure 3.7a, b illustrates an example of count and frequency tables, respectively, computed over the data in Fig. 3.2, reporting the number and percentage of male and female visitors who purchased tickets with a given payment method. Figure 3.7c illustrates an example of magnitude table reporting the average

(a)

	Cash	Check	Credit Card	Debit Card	Tot
M	3	1	1	2	7
F	1	0	2	0	3
Tot	4	1	3	2	10

(b)

	Cash	Check	Credit Card	Debit Card	Tot
M	30	10	10	20	70
F	10	0	20	0	30
Tot	40	10	30	20	100

(c)

	Cash	Check	Credit Card	Debit Card	Tot
M	0	0	12.5	4	16.5
F	0	0	11	3.5	14.5
Tot	0	0	23.5	7.5	31

Fig. 3.7 Count (**a**), frequency (**b**), and magnitude (**c**) tables. (**a**) Number of male and female visitors purchasing tickets with a given payment method. (**b**) Percentage of male and female visitors purchasing tickets with a given payment method. (**c**) Average delay (number of days) between ticket purchase and collection

delay between the purchase of a ticket and its collection. Columns in the tables represent the payment methods, while rows represent male and female visitors, respectively.

Although macrodata do not explicitly include information specifically related to single respondents, sensitive information can still be leaked. To counteract the risk of unintended information disclosure, it is necessary to first identify and then protect cells that can be considered sensitive [14, 23].

Identifying Sensitive Cells Sensitive cells can be identified according to different strategies [23]. In count and frequency tables, sensitive cells can be identified through the *threshold* rule, which classifies a cell as sensitive if its value is less than a given threshold. As an example, consider the macrodata table in Fig. 3.7a and suppose that the threshold is set to 1. In this case, the second and third cells in the first row and the first cell in the second row should be considered sensitive. In magnitude tables, sensitive cells can be identified through different rules (e.g., (n, k)-rule, p-percent rule, pq-rule) all aimed at identifying cells whose value could be exploited to estimate too accurately the contribution of one specific respondent. As an example, according to the (n, k)-rule, a cell is considered sensitive if less than n respondents contribute to more than k % of its value. For instance, the third cell of the first row in Fig. 3.7c does not satisfy (2,90 %)-rule as one respondent only contributes to 100 % of the cell content.

Protecting Sensitive Cells Once detected, sensitive cells must be protected. Several protection techniques have been proposed for macrodata tables. For count and frequency tables, the easiest solution consists in suppressing sensitive cells (*primary suppression*). Unfortunately, primary suppression might open the door to inferences: if marginal totals are published together with the released table, or are publicly known, it might still be possible to restrict the uncertainty about the missing values. To overcome this risk, additional cells need to be suppressed (*secondary suppression*), and linear programming techniques are typically adopted to minimize the number of cells undergoing secondary suppression. Besides suppression, rounding techniques can also be used, which consist in modifying the original value of a cell by rounding it to a near multiple of a chosen base number. The *roll-up categories* technique instead modifies the original table combining rows and/or columns to obtain a less detailed table. A widely used protection technique is *sampling*, which consists in computing the aggregate values in the macrodata table over a representative sample of the collected data (e.g., in our running example over a sample of the museum visitors). Protection is provided by uncertainty, since a recipient does not know whether a target respondent has been considered in the sampling. These protection techniques can be adopted to protect both count and frequency tables. We note however that other more sophisticated approaches have also been proposed to protect sensitive cells in macrodata release.

3.4.2 Protecting Microdata

Many scenarios require that the specific stored data (microdata) be released. Figures 3.2 and 3.3 represent two examples of microdata tables. Although microdata provide higher flexibility and utility for final recipients than macrodata, they are subject to a greater risk of privacy breaches. In particular, a microdata table must be protected against both *identity disclosure* (i.e., disclosure of respondents' identities) and *attribute disclosure* (i.e., disclosure of respondents' sensitive information). In the remainder of this section, we present some well-known approaches to protect microdata tables against identity and attribute disclosures.

3.4.2.1 Identity Disclosure

The attributes in a microdata table can be classified in four classes: *identifiers*, *quasi-identifiers*, *sensitive* attributes, and *nonsensitive* attributes. Identifiers are attributes whose values univocally identify respondents, such as social security numbers and phone numbers. Quasi-identifiers are attributes that can be linked to external sources of information to reduce the uncertainty over the identity of respondents, such as `ZIP`, `DoB`, and `Sex`. Sensitive (nonsensitive, resp.) attributes correspond to the remaining sensitive (nonsensitive, resp.) information of the microdata table. The first step for protecting a microdata table consists in removing (or encrypting) explicit identifiers. A de-identified microdata table, however, does not provide any guarantee of anonymity, since quasi-identifiers might be linked to publicly available information to reidentify respondents. For instance, the de-identified table

(a)

Phone	Name	DoB	Sex	ZIP	TicketType	Payment
		1960/04/10	F	97401	Health	Credit card
		1970/05/12	M	98302	Army	Debit card
		1960/04/04	F	97467	Regular	Cash
		1970/05/20	M	98245	Army	Cash
		1970/07/12	M	98312	Navy	Cash
		1960/04/11	F	97434	Professor	Credit card
		1970/07/25	M	98223	Government	Check
		1970/07/30	M	98389	Marines	Cash
		1970/05/12	M	98290	Army	Credit card
		1950/12/01	*M*	*97210*	*Air Force*	*Debit card*

(b)

Name	Address	City	ZIP	DoB	Sex
...	...	...	...	...	...
John Jacob	1100 Garden State Parkway	Portland	*97210*	*50/12/01*	*male*
...	...	...	...	...	...

Fig. 3.8 An example of de-identified microdata table (**a**) and of publicly available non-de-identified dataset (**b**). (**a**) De-identified version of the relation in Fig. 3.2. (**b**) Portland voters' list

in Fig. 3.8a (computed from the table in Fig. 3.2 removing attributes `Phone` and `Name`) can be linked with the public voters' list of Portland (Fig. 3.8b), which includes a single tuple related to a male, living in the 97,210 area, born on 1 December 1950. This combination of values, if unique in the external world as well, uniquely identifies the corresponding tuple in the microdata table as pertaining to *John Jacob*, 1100 Garden State Parkway, revealing that he works in the Air Force and that he paid the visit to the museum with a debit card. It is interesting to note that a study performed on 2000 US Census data showed that 63 % of the US population can be *uniquely identified* combining their gender, ZIP code, and complete date of birth [27].

To protect respondents' identities from the linking attack illustrated above, k-anonymity [43] requires that any released tuple be *indistinguishably related* to no less than a certain number k of respondents. Since reidentification through linking attacks exploits quasi-identifying attributes, this requirement is translated as follows: *Each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents* [43]. Starting from the assumption that each respondent is represented by a tuple in a microdata table (and, vice versa, that each tuple is related to a single respondent), a microdata table satisfies the k-anonymity requirement iff (1) each tuple in the table cannot be related to less than k individuals in the population and (2) each individual in the population cannot be related to less than k tuples in the table. Since it is not possible to take into consideration all possible external sources of information, the k-anonymity requirement is typically enforced by taking a safe approach and requiring each respondent to be indistinguishable from at least $k - 1$ respondents of the table itself (which represents a sufficient, though not necessary, condition for the k-anonymity requirement). A table is therefore said to be k-anonymous if each

(a)

Phone	Name	DoB	Sex	ZIP	TicketType
		1970/05/**	M	98***	Army
		1970/05/**	M	98***	Army
		1970/05/**	M	98***	Army
		1960/04/**	F	97***	Health
		1960/04/**	F	97***	Regular
		1960/04/**	F	97***	Professor
		1970/07/**	M	98***	Navy
		1970/07/**	M	98***	Government
		1970/07/**	M	98***	Marines

(b)

Phone	Name	DoB	Sex	ZIP	TicketType
		1970/**/**	M	983**	Army
		1970/**/**	M	983**	Marines
		1970/**/**	M	983**	Navy
		1960/**/**	F	974**	Health
		1960/**/**	F	974**	Regular
		1960/**/**	F	974**	Professor
		1970/**/**	M	982**	Army
		1970/**/**	M	982**	Army
		1970/**/**	M	982**	Government

Fig. 3.9 An example of 3-anonymous table (**a**) and 3-anonymous and 2-diverse table (**b**)

combination of values of the quasi-identifier appears with either zero or at least k occurrences in the released table.

k-Anonymity is traditionally enforced by adopting *generalization* and *suppression* techniques on the attributes composing the quasi-identifier, without modifying sensitive and nonsensitive attributes. Generalization substitutes the original values with more general values (e.g., the date of birth can be generalized by releasing only the year of birth). Suppression consists in removing information and is particularly useful to reduce the amount of generalization necessary to guarantee k-anonymity whenever a limited number of outliers (i.e., quasi-identifying values with less than k occurrences) would require considerable generalizations. Generalization and suppression can be applied at different levels of granularity, and several approaches have been proposed combining them in different ways [7, 13, 34, 35, 43]. The majority of available solutions rely on attribute generalization and tuple suppression. Figure 3.9a illustrates a 3-anonymous microdata table obtained from the table in Fig. 3.8, where attribute Payment has been projected out since it is not intended for release. Attributes DoB, Sex, and ZIP in the table are considered as the quasi-identifier, and TicketType is considered sensitive as the museum is not authorized to disclose such information. The 3-anonymous table has been obtained by generalizing attributes DoB (only the year and month of birth are released) and ZIP (only the first two digits are released). Also, the outlier tuple related to John Jacob has been suppressed not to force further generalization on the date of birth, since John is the only respondent born in 1950.

Reducing the details in the anonymized table, k-anonymity inevitably causes information loss. To find a good trade-off between data protection and utility for final recipients, it is necessary to compute a k-anonymous table minimizing the adoption of generalization and suppression. To this end, both exact and heuristic algorithms can be adopted [13].

3.4.2.2 Attribute Disclosure

k-Anonymity, while effective for protecting respondents' identities, does not protect against attribute disclosure. To protect the association between respondents' identities and their values of sensitive attributes, alternative solutions extending k-anonymity have been proposed. In the following, we will illustrate ℓ-diversity and t-closeness, two well-known extensions that counteract attribute disclosure.

ℓ-Diversity ℓ-Diversity has been proposed to counteract two specific attacks that might cause attribute disclosure in a k-anonymous table, namely, the *homogeneity attack* [37, 43] and the *external knowledge attack* [37].

- *Homogeneity attack*. k-Anonymity does not impose restrictions on the values that can be assumed by the sensitive attribute in an equivalence class (i.e., by the tuples sharing the same value for the quasi-identifier). As a consequence, it might happen that a given equivalence class includes tuples with the same sensitive value. If a data recipient knows the quasi-identifier value of an individual that is represented in the table, the data recipient can identify the equivalence class corresponding to the target respondent and then infer the value of her/his sensitive attribute. For instance, consider the 3-anonymous table in Fig. 3.9a and suppose that a recipient knows that *Daniel* born on 20 May 1970 is included in the table. Since all the tuple in the equivalence class with quasi-identifier value equal to (1970/05/**,M,98***) have *army* as value for attribute `TicketType`, the recipient can infer that *Daniel* works in the army, which represents a sensitive information not intended for disclosure in our example.
- *External knowledge attack*. k-Anonymity assumes that the only external knowledge a recipient can have be represented by external sources linking respondents' quasi-identifier values to their identities. However, a recipient might exploit some additional external knowledge about some respondents to infer their associated sensitive information. For instance, consider a 3-anonymous equivalence class where two out of three tuples have *army* as value for attribute `TicketType`, while the third tuple has value *health*. Suppose now that a recipient knows that a target respondent *Phil* is included in this equivalence class and that *Phil* does not suffer from any specific disease. The recipient can easily infer that *Phil* is not likely to pay for a reduced ticket for medical conditions, hence discovering that he works in the army.

To counteract these two attacks, ℓ-diversity extends k-anonymity by requiring the existence of at least ℓ *well-represented* values for the sensitive attribute in each equivalence class [37]. A straightforward understanding of "well-represented" values requires each equivalence class to have *at least* ℓ different values for the sensitive attribute. For instance, the 3-anonymous table in Fig. 3.9b is also 2-diverse. It is easy to see that an ℓ-diverse table is not vulnerable to the homogeneity attack as each equivalence class has at least ℓ different values for the sensitive attribute. Also, external knowledge attacks lose effectiveness as ℓ increases, since more external knowledge is necessary to associate a specific sensitive attribute value with a target respondent.

An ℓ-diverse table that minimizes the adoption of generalization and suppression to reduce information loss can be computed using any algorithm that computes an optimal k-anonymous table by simply adding a control to check whether the condition on the diversity of the sensitive attribute values is satisfied by all the equivalence classes in the table [37].

***t*-Closeness** An ℓ-diverse table might still cause improper disclosures of sensitive information, since it is vulnerable to the following two attacks [36]:

- *Skewness attack*. This attack may occur when the distribution of values of the sensitive attribute within a given equivalence class differs from the general (demographic or in the whole table) one. Indeed, differences in these distributions highlight changes in the probability with which a respondent in the equivalence class is associated with a specific sensitive value. As an example, the 2-diverse table in Fig. 3.9b leaks the information that respondents in the third equivalence class work in the army with 2/3 probability, compared to the 1/3 probability over the whole relation.
- *Similarity attack*. This attack may occur when the values of the sensitive attribute within a given equivalence class are (despite being syntactically different as demanded by ℓ-diversity) semantically similar. For instance, all respondents in the first equivalence class of the 2-diverse table in Fig. 3.9b work in the armed forces, as the values assumed by the three tuples are *army*, *marines*, and *navy*.

To counteract these two attacks, t-closeness extends the k-anonymity requirement taking into account the distribution of sensitive values in equivalence classes [36]. t-Closeness requires that the frequency distribution of the sensitive values in each equivalence class be close (i.e., with distance smaller than a fixed threshold t) to the distribution of the same attribute values in the microdata table. Note that the distance between the frequency distributions of the sensitive attribute values in the released table and in each equivalence class can be evaluated adopting several metrics (e.g., Earth Mover Distance [36]). The enforcement of the t-closeness requirement makes the skewness attack harmless, as the knowledge of the quasi-identifier value for a target respondent does not change the probability of inferring the sensitive value associated with her/him. t-Closeness also reduces the effectiveness of the similarity attack: the presence of semantically similar values in an equivalence class can only be due to the presence of the same values in the whole microdata table.

3.4.3 Protecting Data Streams

The solutions proposed to provide k-anonymity, ℓ-diversity, and t-closeness, as well as the majority of microdata protection techniques, assume all data that need to be released to be available at initial time. Then, the chosen protection technique can be applied on the whole collection at once. In the context of pervasive systems, however, this assumption might be too strong as new data are continuously generated (and possibly need to be immediately released), forming a so-called *data stream*. In the context of data streams, timeliness usually assumes paramount importance in the release process, as disclosing old or outdated data is likely to be of little interest to the final recipients. Data streams can be protected by applying ad hoc solutions to guarantee k-anonymity, which are typically based on generalization and on the introduction of a limited delay in data publication. The first solution in this direction has been proposed in [50] and consists in publishing all the tuples in an equivalence class at the same time. To this end, a set of equivalence classes—all initially empty—is prepared. When a new tuple is generated by the stream, it is inserted into a suitable equivalence class, if such a class exists; a new equivalence class suitable for the tuple is generated otherwise. As soon as an equivalence class includes k tuples (which must be related to k different respondents), these tuples are generalized to the same quasi-identifier value and published.

Aiming at enforcing ℓ-diversity, rather than k-anonymity, an alternative approach has been proposed in [46], where data are assumed to be generated and published as "snapshots" (i.e., sets of records available at a given moment of time) of d tuples each. This technique combines traditional generalization and suppression techniques with *tuple relocation* to guarantee ℓ-diversity. In a nutshell, relocation consists in moving a tuple from one snapshot to a more recent one if this delay in data publishing can be useful to satisfy ℓ-diversity.

3.5 Privacy-Preserving Data Storage

Privacy concerns can also arise when data storage and management is delegated (for various reasons, such as economical costs) to external, possibly not fully trusted, storage providers. These scenarios present several challenging issues, ranging from fault tolerance, data protection, and data and query integrity to private access (e.g., [31, 32, 44]). Relying on external providers is particularly appealing in the context of pervasive data, due to the high volume of data generated requiring large storage space that, for example, the museum is most likely not to have. In this scenario, to protect confidentiality of data from unauthorized users—including the external provider—a straightforward solution is represented by wrapping an encryption layer around the data to be protected. While effective for protecting data confidentiality, encryption inevitably complicates query execution that becomes possible only with the adoption of expensive ad hoc encryption schemes [10, 18, 26, 47] or indexes [9, 19, 29, 45]. Moreover, in many scenarios, the sensitive information to be protected is represented by the association among data items, rather than the data themselves singularly taken. For instance, with reference to our running example, knowing that a user named *Alice* visited the museum and that a user paid a reduced ticket for health reasons may not be sensitive. But discovering that *Alice*, who visited the museum, paid a reduced ticket because of her health problems might represent a confidential information.

Sensitive associations can be modeled as *confidentiality constraints*, which are a set of attributes whose joint visibility (i.e., association) is sensitive. Attributes whose values are sensitive per se correspond to singleton constraints. For instance, with

Fig. 3.10 Confidentiality constraints for the relation in Fig. 3.2

c_1 = {Phone}
c_2 = {Name, Ticket Type}
c_3 = {Name, Payment}
c_4 = {Ticket Type, Payment}

reference to our running example, Fig. 3.10 represents an example of confidentiality constraints over the relation in Fig. 3.2. Constraint c_1 states that the phone numbers of the visitors represent sensitive information to be protected, and constraint c_2 (c_3 and c_4, respectively) states that the association between visitors' name and type of ticket (name and payment and type of ticket and payment, respectively) is sensitive and must be protected.

The adoption of encryption to satisfy confidentiality can be (partially) avoided storing the collected data through a set of *privacy-preserving views*, which are defined in such a way as to satisfy confidentiality constraints [1, 12, 15–17]. To this end, sensitive associations among attributes are broken (fragmented) by storing the attributes composing each of them in different views. Sensitive associations are then protected by restricting visibility over the views or by ensuring their unlinkability.

Given a relation to be protected, privacy-preserving views can be defined according to different paradigms, differing on how data are fragmented to satisfy the confidentiality constraints. In the following, we briefly illustrate the three most important approaches that can be used in our scenario to create privacy-preserving views.

Two Can Keep a Secret [1] Given a data collection, this strategy produces two views V_1 and V_2 to be stored at two noncommunicating providers. Sensitive attributes are protected by *obfuscating* (e.g., encrypting) them, while sensitive associations are protected by distributing the attributes in the confidentiality constraint between the two views. In addition to sensitive attributes, some attributes also appearing in sensitive associations might be obfuscated when two views are not sufficient to protect all sensitive associations. A common attribute `tid` is included in both views, to allow the data owner (and all authorized users) to reconstruct the original relation. Figure 3.11a illustrates two views defined over the relation in Fig. 3.2 satisfying the constraints in Fig. 3.10. Note that attribute `Payment`, although not sensitive per se, has been obfuscated in both views: in fact, its plaintext representation in view V_1 would violate constraint c_3 and in view V_2 would violate c_4.

Multiple Views [15, 17] Given a data collection, this strategy produces a set $\{V_1, \ldots, V_n\}$ of unlinkable views. The multiple views approach removes the limiting assumption of the existence of two noncommunicating providers, hence it results applicable to several real-world scenarios. According to this approach, sensitive attributes are protected with encryption, while sensitive associations are protected by distributing their attributes in different views. Views include disjoint sets of attributes to guarantee their unlinkability. Note that since the number of views that can be produced is not limited to two, no attribute that is not sensitive per se needs to be protected with encryption. To allow query execution over a single view, each view

(a) V_1

tid	Name	DoB	Sex	Payment^k	Phone^k
t_1	Alice	1960/04/10	F	α	λ
t_2	Bob	1970/05/12	M	β	μ
t_3	Carol	1960/04/04	F	γ	ν
t_4	Daniel	1970/05/20	M	δ	ξ
t_5	Erik	1970/07/12	M	ε	o
t_6	Fred	1960/04/11	F	ζ	π
t_7	Greg	1970/07/25	M	η	ρ
t_8	Hal	1970/07/30	M	θ	σ
t_9	Ian	1970/05/12	M	ι	τ
t_{10}	John	1950/12/01	M	κ	υ

V_2

tid	TicketType	ZIP	Payment^k	Phone^k
t_1	Health	97401	ϕ	Γ
t_2	Army	98302	χ	Δ
t_3	Regular	97467	ψ	Θ
t_4	Army	98245	ω	Λ
t_5	Navy	98312	ε	Ξ
t_6	Professor	97434	ϑ	Π
t_7	Government	98223	ϖ	Σ
t_8	Marines	98389	ρ	Υ
t_9	Army	98290	ς	Φ
t_{10}	Air Force	97210	φ	Ψ

(b) V_1

salt	enc	Name	DoB
s_{01}	α	Alice	1960/04/10
s_{02}	β	Bob	1970/05/12
s_{03}	γ	Carol	1960/04/04
s_{04}	δ	Daniel	1970/05/20
s_{05}	ε	Erik	1970/07/12
s_{06}	ζ	Fred	1960/04/11
s_{07}	η	Greg	1970/07/25
s_{08}	θ	Hal	1970/07/30
s_{09}	ι	Ian	1970/05/12
s_{10}	κ	John	1950/12/01

V_2

salt	enc	TicketType	Sex
s_{11}	λ	Health	F
s_{12}	μ	Army	M
s_{13}	ν	Regular	F
s_{14}	ξ	Army	M
s_{15}	o	Navy	M
s_{16}	π	Professor	F
s_{17}	ρ	Government	M
s_{18}	υ	Marines	M
s_{19}	ϕ	Army	M
s_{20}	χ	Assistant	M

V_3

salt	enc	Payment	ZIP
s_{21}	ψ	Credit card	97401
s_{22}	o	Debit card	98302
s_{23}	π	Cash	97467
s_{24}	ρ	Cash	98245
s_{25}	σ	Cash	98312
s_{26}	τ	Credit card	97434
s_{27}	υ	Check	98223
s_{28}	ϕ	Cash	98389
s_{29}	χ	Credit card	98290
s_{30}	ψ	Debit card	97210

(c) V_o

Phone	Name	TicketType
(800) 917-5551	Alice	Health
(500) 234-5678	Bob	Army
(541) 271-2136	Carol	Regular
(360) 474-4614	Daniel	Army
(360) 373-2030	Erik	Navy
(541) 946-1711	Fred	Professor
(360) 435-3746	Greg	Government
(253) 863-5555	Hal	Marines
(360) 794-7058	Ian	Army
(503) 497-91 33	John	Assistant

V_s

DoB	Sex	ZIP	Payment
1960/04/10	F	97401	Credit card
1970/05/12	M	98302	Debit card
1960/04/04	F	97467	Cash
1970/05/20	M	98245	Cash
1970/07/12	M	98312	Cash
1960/04/11	F	97434	Credit card
1970/07/25	M	98223	Check
1970/07/30	M	98389	Cash
1970/05/12	M	98290	Credit card
1950/12/01	M	97210	Debit card

Fig. 3.11 Privacy-preserving views over the relation in Fig. 3.2 satisfying the constraints in Fig. 3.10. (**a**) Two can keep a secret. (**b**) Multiple views. (**c**) Keep a few

is *complete*, meaning that it stores all the attributes of the original relation in either encrypted or plaintext form. Attributes that are encrypted in a view are encrypted in a single encrypted chunk (at the level of tuple), which is properly "salted" not to expose the frequencies of values. Figure 3.11 illustrates three views defined over the relation in Fig. 3.2 satisfying the constraints in Fig. 3.10.

Keep a Few [16] Given a data collection, this strategy produces two views V_o and V_s only, one of which (i.e., V_o) is stored at a trusted party (e.g., the data owner). The *keep a few* approach completely departs from encryption: sensitive attributes are protected by storing them in V_o maintained at the trusted party, while sensitive associations are protected by storing at least one attribute, for each association, in V_o. The two views include a common attribute tid to allow the owner and authorized users to reconstruct the content of the original relation. Figure 3.11c illustrates the two views V_o and V_s defined over the relation in Fig. 3.2 satisfying the constraints in Fig. 3.10, where view V_s stores attribute Phone, which is sensitive per se, and one attribute for constraints c_2, c_3, and c_4.

3.6 Conclusions

The pervasive availability of computing infrastructures, often enriched with sensorial capabilities and context awareness to provide personalized services to users, causes unprecedented privacy risks that need to be carefully tackled. In this chapter, starting from a sample scenario, we have illustrated such privacy risks and discussed some available solutions to counteract them when accessing, sharing, and storing information collected through pervasive systems.

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: Proceedings of CIDR 2005, Asilomar, CA (2005)
2. Ardagna, C., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: Proceedings of DBSec 2007, Redondo Beach, CA (2007)
3. Ardagna, C., Jajodia, S., Samarati, P., Stavrou, A.: Providing mobile users' anonymity in hybrid networks. In: Proceedings of ESORICS 2010, Athens (2010)
4. Ardagna, C., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: An obfuscation-based approach for protecting location privacy. IEEE Trans. Dependable Secure Comput. **8**(1), 13–27 (2011)
5. Ardagna, C., Livraga, G., Samarati, P.: Protecting privacy of user information in continuous location-based services. In: Proceedings of CSE 2012, Paphos (2012)
6. Ardagna, C., Jajodia, S., Samarati, P., Stavrou, A.: Providing users' anonymity in mobile hybrid networks. ACM Trans. Internet Technol. **12**(3), 1–33, Article 7 (2013)
7. Bayardo, R.J., Agrawal, R.: Data privacy through optimal *k*-anonymization. In: Proceedings of ICDE 2005, Tokyo (2005)
8. Bettini, C., Jajodia, S., Samarati, P., Wang, X.S. (eds.): Privacy in Location-Based Applications: Introduction, Research Issues and Applications. Lecture Notes in Computer Science, vol. 5599. Springer, Berlin (2009)
9. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Trans. Inf. Syst. Secur. **8**(1), 119–152 (2005)
10. Chang, Y., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Proceedings of ACNS 2005, New York, NY (2005)
11. Chow, C.Y., Mokbel, M.: Enabling private continuous queries for revealed user locations. In: Proceedings of SSTD 2007, Boston, MA (2007)

12. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS 2007), Dresden (2007)
13. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-Anonymity. In: Yu, T., Jajodia, S. (eds.) Secure Data Management in Decentralized Systems. Springer, Berlin (2007)
14. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: Microdata protection. In: Yu, T., Jajodia, S. (eds.) Secure Data Management in Decentralized Systems. Springer, Berlin (2007)
15. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. **13**(3), 22:1–22:33 (2010)
16. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. J. Comput. Secur. **19**(3), 531–566 (2011)
17. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: An OBDD approach to enforce confidentiality and visibility constraints in data publishing. J. Comput. Secur. **20**(5), 463–508 (2012)
18. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of CCS 2006, Alexandria, VA (2006)
19. Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proceedings of ACM CCS 2003, Washington, DC (2003)
20. De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: Protecting privacy in data release. In: Aldini, A., Gorrieri, R. (eds.) Foundations of Security Analysis and Design VI. Springer, New York (2011)
21. De Capitani di Vimercati, S., Foresti, S., Samarati, P.: Managing and accessing data in the cloud: privacy risks and approaches. In: Proceedings of CRiSIS 2012, Cork (2012)
22. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Proceedings of PERVASIVE 2005, Munich (2005)
23. Federal Committee on Statistical Methodology: Statistical policy working paper 22. Report on Statistical Disclosure Limitation Methodology (1994)
24. Gamassi, M., Piuri, V., Sana, D., Scotti, F.: Robust fingerprint detection for access control. In: Proceedings of RoboCare 2005, Rome (2005)
25. Gedik, B., Liu, L.: Protecting location privacy with personalized *k*-anonymity: architecture and algorithms. IEEE Trans. Mob. Comput. **7**(1), 1–18 (2008)
26. Goh, E.J.: Secure indexes. Technical Report 2003/216, Cryptology ePrint Archive. Http://eprint.iacr.org/ (2003)
27. Golle, P.: Revisiting the uniqueness of simple demographics in the US population. In: Proceedings of WPES 2006, Alexandria, VA (2006)
28. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of MobiSys 2003, San Francisco, CA (2003)
29. Hacigümüş, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: Proceedings of ICDE 2002, San Jose, CA (2002)
30. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: Proceedings of CSE 2012, Paphos (2012)
31. Jhawar, R., Piuri, V., Santambrogio, M.: A comprehensive conceptual system-level approach to fault tolerance in cloud computing. In: Proceedings of SysCon 2012, Vancouver, BC (2012)
32. Jhawar, R., Piuri, V., Santambrogio, M.: Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst. J. **7**(2), 288–297 (2013)
33. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing location-based identity inference in anonymous spatial queries. IEEE Trans. Knowl. Data Eng. **19**(12), 1719–1733 (2007)
34. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient full-domain *k*-anonymity. In: Proceedings of SIGMOD 2005, Baltimore, MD (2005)

35. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian multidimensional *k*-anonymity. In: Proceedings of ICDE 2006, Atlanta, GA (2006)
36. Li, N., Li, T., Venkatasubramanian, S.: *t*-closeness: privacy beyond *k*-anonymity and ℓ-diversity. In: Proceedings of ICDE 2007, Istanbul (2007)
37. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: privacy beyond *k*-anonymity. ACM Trans. Knowl. Discovery Data **1**(1), 3:1–3:52 (2007)
38. Meyerowitz, J., Roy Choudhury, R.: Hiding stars with fireworks: location privacy through camouflage. In: Proceedings of MobiCom 2009, Beijing (2009)
39. Mokbel, M., Chow, C.Y., Aref, W.: The new Casper: query processing for location services without compromising privacy. In: Proceedings of VLDB 2006, Seoul (2006)
40. Mouratidis, K., Yiu, M.: Anonymous query processing in road networks. IEEE Trans. Knowl. Data Eng. **22**(1), 2–15 (2010)
41. Pan, X., Meng, X., Xu, J.: Distortion-based anonymity for continuous queries in location-based mobile services. In: Proceedings of ACM GIS 2009, Seattle, WA (2009)
42. Piuri, V., Scotti, F.: Fingerprint biometrics via low-cost sensors and webcams. In: Proceedings of BTAS 2008, Washington, DC (2008)
43. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. **13**(6), 1010–1027 (2001)
44. Samarati, P.: Data security and privacy in the cloud. In: Proceedings of ISPEC 2014, Fuzhou (2014)
45. Samarati, P., De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of ASIACCS 2010, Beijing (2010)
46. Wang, K., Xu, Y., Wong, R., Fu, A.: Anonymizing temporal data. In: Proceedings of ICDM 2010, Sydney (2010)
47. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1467–1479 (2012)
48. Weiser, M., Gold, R., Brown, J.: The origins of ubiquitous computing research at parc in the late 1980s. IBM Syst. J. **38**(4), 693–696 (1999)
49. Xu, T., Cai, Y.: Location anonymity in continuous location-based services. In: Proceedings of ACM GIS 2007, Seattle, WA (2007)
50. Zhou, B., Han, Y., Pei, J., Jiang, B., Tao, Y., Jia, Y.: Continuous privacy preserving publishing of data streams. In: Proceedings of the EDBT 2009, Saint Petersburg (2009)

Part II
Sensors, Data Streams, and Storage

A complete pervasive application is composed of several architectural layers requiring different abilities, from low-level hardware programming to the design of high-level abstractions.

Part II mainly focuses on the fundamental technological issues beneath a pervasive data management system.

In Chap. 4, the different devices composing a WSN are introduced and described by means of an abstract conceptual model. This allows to query and send commands to the devices using a single database-style language. Examples will be shown using PERvasive LAnguage (PerLa), a declarative structured query language (SQL)-like language and infrastructure for data management in pervasive systems, mainly oriented to monitoring applications but, with its context management extension, also suitable as a support substratum for the deployment of autonomic systems.

The following two chapters discuss the main approaches to information flow processing: the first, introduced in Chap. 5, is more related to the viewpoint of the database research community (data stream management systems (DSMSs)); the other, introduced in Chap. 6, looks at the problem from the software engineering perspective and is more related to the distributed systems community (complex event processing (CEP) systems). The main differences among the two paradigms lie in the different ways of considering the stream:

- In DSMS, the stream is seen as an infinite set of timestamped tuples, ordered by their own timestamp. In some advanced implementation, tuples could also be annotated with an expiration time that lets the system know when a tuple has to be discarded.
- In CEP, a stream represents an infinite sequence of events. Each event is characterized by an identifier of its type and a timestamp; the CEP paradigm provides event consumption policies to let the system discard an event (or a set of events) after having used them for detecting a given sequence rather than using an event expiration time.

While DSMSs propose an SQL-like query paradigm, trying to keep those systems as similar as possible to traditional relational databases, CEPs are more like traditional publish/subscribe systems, usually allowing querying by means of logical rules. None of the two approaches exclude the other; scenarios exist in which one performs better than the other: CEPs usually perform better in fields like surveillance, while DSMSs are better suited for data analysis.

Chapter 7 addresses the problem of data integration in data streams; this requires a new generation of data integration solutions tailored to meet the requirements. Often separate organizations and, even, departments of the same organization collect and manage data independently creating so-called data silos, which must be bridged and managed through a uniform query interface. Two extreme solutions to data integration—data driven and query driven—are considered.

Chapter 4
Sensors and Wireless Sensor Networks as Data Sources: Models and Languages

Fabio A. Schreiber and Manuel Roveri

4.1 Introduction

Wireless sensor networks (WSNs) are a clear example of the disappearing technology expressed by Mark Weiser [49] since they allow humans to interact with the environment, feeding the user with a large set of heterogeneous data. At the same time, the complexity of the envisaged WSN-based systems grew from a handful of homogeneous sensors to hundreds or thousands of devices differing as to their capabilities, technologies, architectures, and languages. The result of this inflationary expansion is the difficulties an application programmer encounters, mainly in dealing with the languages and protocols, which characterize different portions of the system, and those encountered by a system designer in optimizing sampling, storage, and transmission strategies in order to save as much as possible power in the batteries, which are the most critical components in a system of mostly unattended small devices.

Thus, like in other domains of computer systems design, we need models that abstract the structure and behavior of the WSN and of its components at different levels in order to allow the designer to solve problems in a disciplined way. A complete pervasive application is composed of several architectural layers requiring different abilities, from low-level hardware programming to the design of high-level abstractions; physical integration is achieved operating on low-level layers, while spontaneous interoperability is mostly related to the highest levels.

Moreover, if considered together, the large mass of data produced by a WSN could potentially generate confusion. Thus, filtering this plethora of data according to the user's needs and, more importantly, their current situation is the key for

F.A. Schreiber (✉) • M. Roveri
DEIB, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
e-mail: fabio.schreiber@polimi.it; manuel.roveri@polimi.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric
Systems and Applications, DOI 10.1007/978-3-319-20062-0_4

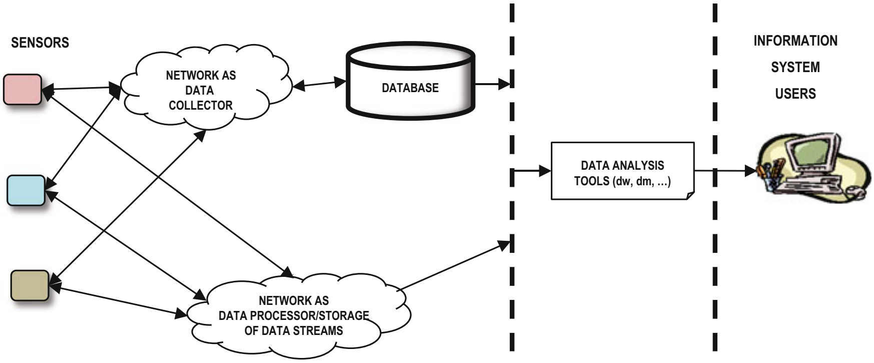

SENSORS
NETWORK AS DATA COLLECTOR
DATABASE
DATA ANALYSIS TOOLS (dw, dm, ...)
INFORMATION SYSTEM USERS
NETWORK AS DATA PROCESSOR/STORAGE OF DATA STREAMS

Fig. 4.1 Data-centric views of a WSN

effective and efficient generation of knowledge. Merging modern pervasive system frameworks with the concept of context is a fundamental cornerstone that must be achieved. Besides producing the data required by the applications, the sensors of the network can be used to characterize and discover possible contexts and for defining suitable actions accordingly, as discussed in Chap. 12.

This chapter mainly focuses on the issues related to the heterogeneity of the different devices composing a WSN: this aspect is investigated both at physical integration and at data management levels. We shall give a "database-style" view of the WSN providing conceptual models both for the system and its components.

In Fig. 4.1, as described in [11], two approaches are shown for modeling a WSN from a data management point of view: the first approach, in the upper part of the figure, considers the network just as a data-gathering medium which collects data produced by sensors and stores them into a traditional database, which is then queried by the application software by means of standard database languages (e.g., SQL) and possibly processed for building an associate data warehouse; the second one, in the lower part, considers the network as an active processor/(temporary) storage of raw or value-added data streams produced by sensors, which are directly accessed by the application software through a data stream management system (DSMS).

Examples will be shown using PerLa [44], a declarative SQL-like language and infrastructure for data management in pervasive systems, mainly oriented to monitoring applications but, with its context management extension, also suitable as a support substratum for the deployment of autonomic systems [45].

4.2 Sensors

The first layer of the data-centric view of WSNs shown in Fig. 4.1 consists of the sensors, whose goal is to measure a physical quantity from the environment and transform it into a signal (either analog or digital) that drives outputs/actuators or

Table 4.1 Examples of possible applications, physical phenomena to be monitored, and the corresponding sensors

Applications	Physical phenomenon	Sensors
Environmental monitoring	Temperature	Thermometers, thermocouples, thermistor
	Light	Photodiodes/phototransistor, CCD cameras
	Humidity	Humidity sensors
	Pressure	Switches, strain gauges
Visual and audio processing	Visual	CCD and other cameras
	Audio	Microphones
Motion/acceleration analysis	Motion/acceleration	Accelerometers, motion detectors, magnetic fields, angular sensors
Location	Indoor	Tags, badges
	Outdoor	GPS, GSM cells
Biological and chemical analysis	Biological	ECG, heart rate sensors, blood pressure sensors, pulse oximetry, skin resistance, DNA analyzer
	Chemical	Contaminant sensors, solid/liquid/gas sensors, electrochemical cells

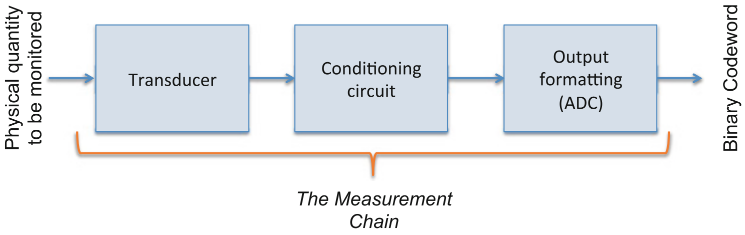


Fig. 4.2 The measurement chain of sensors

feeds a data analysis tool [24]. The choice of a sensor in a WSN is application dependent and strictly related to the physical phenomenon object of the monitoring action. The wide and heterogeneous range of possible physical phenomena to be monitored is well reflected by the complexity and the heterogeneity of the sensors that have been designed and developed to acquire data from them. Examples of possible applications, physical phenomena to be monitored, and the corresponding sensors of interest are summarized in Table 4.1.

The measurement of a physical quantity and the subsequent conversion into signals are performed by sensors through a measurement chain that is generally composed of three different stages [6, 23, 24], as in Fig. 4.2: transducer, conditioning circuit, and output formatting.

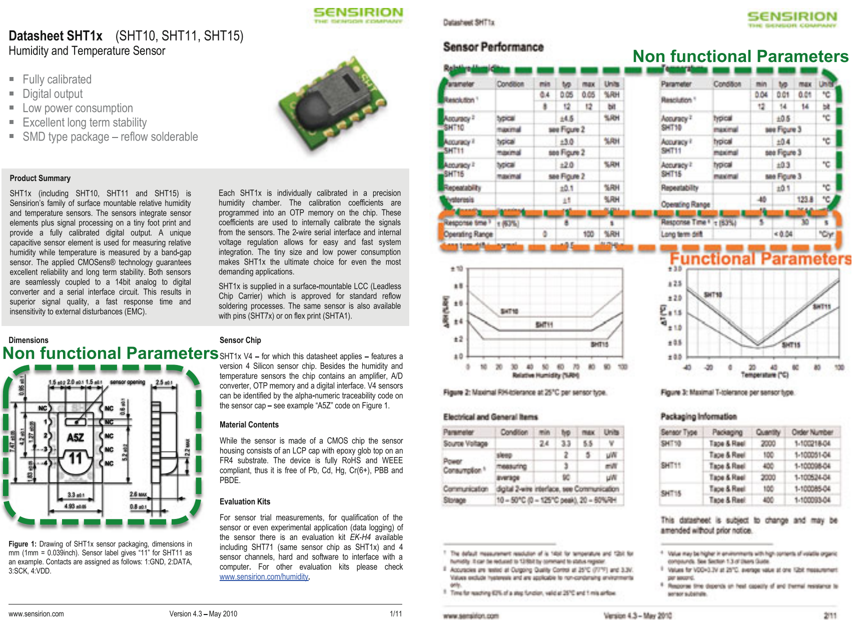
The transducer is the device that transforms one form of energy into another, here converting the monitored physical quantity into an electric signal. The following stage conditions the signal provided by the transducer to reduce the effect of noise, amplify the sensitivity of the sensor, and adapt the electrical properties of the signal to the needs of the next stage. The final output formatting stage aims at transforming the conditioned signal into a signal ready to be used to drive outputs or actuators and to be processed by the data analysis tool. In case of digital-output sensors, which is the case we are considering here, the output formatting stage refers to the analog-to-digital conversion (ADC) that converts the conditioned signal into a codeword represented in binary format. Further stages could be considered as well to correct or calibrate data at the digital level (i.e., improving the quality of the binary-formatted data by introducing corrections or error compensations).

As stated in Sect. 4.1, WSNs are characterized by a heterogeneity of devices making the design and management of these systems particularly difficult. From the sensing point of view, the reason of this heterogeneity is twofold. First, different kinds of sensors can be considered within the same application to provide different views of different, but related, physical phenomena. For example, monitoring the quality of an environment (e.g., assess the quality of the environmental parameters within a room or a building) would require to consider several different physical phenomena like temperature, humidity, light, radiation, and contaminant concentration, each requiring a specific kind of sensor devoted to monitor it. Second, the operating mode of each kind of sensor is characterized by its own parameters (e.g., the sampling frequency, the output format, the power supply). Both reasons led to meaningfully increase the heterogeneity of the envisaged WSN-based system, and this aspect must be taken into account by WSN application designers.

The measurement chain of sensors is characterized by both functional and nonfunctional attributes specifying the functional behavior and the requirements/properties of the sensor itself, respectively. Functional attributes refer to specific functions or behaviors the sensor must exhibit. Examples of functional attributes are the measurement range (i.e., the range of measurements the sensor can acquire), the duty-cycle sampling rates (i.e., the range of sampling rates the sensor can support), and the condition stage parameters (e.g., the filter parameters). Differently, nonfunctional attributes refer to properties characterizing how the sensor is providing its functionalities. Examples of nonfunctional attributes are the size/weight of the sensor, the accuracy and the precision of the measurement chain, and the power consumption.

Functional and nonfunctional attributes of a sensor are generally detailed in the sensor documentation provided by the manufacturer, i.e., the sensor datasheet. A datasheet of a real humidity and temperature sensor (i.e., SENSIRION SHT1x) is shown in Fig. 4.3, where functional and nonfunctional attributes are highlighted.

The heterogeneity makes the management of data complex since (at least in principle) a procedure involving each specific sensor should be considered. To address this problem, the data-centric view approach based on declarative querying (presented in Fig. 4.1) allows to hide the heterogeneity of devices to the application designers by means of sensor attribute abstraction. To achieve this goal, both


SENSIRION
THE SENSOR COMPANY

Datasheet SHT1x (SHT10, SHT11, SHT15)
Humidity and Temperature Sensor

- Fully calibrated
- Digital output
- Low power consumption
- Excellent long term stability
- SMD type package – reflow solderable

Product Summary

SHT1x (including SHT10, SHT11 and SHT15) is Sensirion's family of surface mountable relative humidity and temperature sensors. The sensors integrate sensor elements plus signal processing on a tiny foot print and provide a fully calibrated digital output. A unique capacitive sensor element is used for measuring relative humidity while temperature is measured by a band-gap sensor. The applied CMOSens® technology guarantees excellent reliability and long term stability. Both sensors are seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC).

Each SHT1x is individually calibrated in a precision humidity chamber. The calibration coefficients are programmed into an OTP memory on the chip. These coefficients are used to internally calibrate the signals from the sensors. The 2-wire serial interface and internal voltage regulation allows for easy and fast system integration. The tiny size and low power consumption makes SHT1x the ultimate choice for even the most demanding applications.

SHT1x is supplied in a surface-mountable LCC (Leadless Chip Carrier) which is approved for standard reflow soldering processes. The same sensor is also available with pins (SHT7x) or on flex print (SHTA1).

Dimensions

Non functional Parameters

Figure 1: Drawing of SHT1x sensor packaging, dimensions in mm (1mm = 0.039inch). Sensor label gives "11" for SHT11 as an example. Contacts are assigned as follows: 1:GND, 2:DATA, 3:SCK, 4:VDD.

Sensor Chip

SHT1x V4 – for which this datasheet applies – features a version 4 Silicon sensor chip. Besides the humidity and temperature sensors the chip contains an amplifier, A/D converter, OTP memory and a digital interface. V4 sensors can be identified by the alpha-numeric traceability code on the sensor cap – see example "A5Z" code on Figure 1.

Material Contents

While the sensor is made of a CMOS chip the sensor housing consists of an LCP cap with epoxy glob top on an FR4 substrate. The device is fully RoHS and WEEE compliant, thus it is free of Pb, Cd, Hg, Cr(6+), PBB and PBDE.

Evaluation Kits

For sensor trial measurements, for qualification of the sensor or even experimental application (data logging) of the sensor there is an evaluation kit *EK-H4* available including SHT71 (same sensor chip as SHT1x) and 4 sensor channels, hard and software to interface with a computer. For other evaluation kits please check www.sensirion.com/humidity.

www.sensirion.com Version 4.3 – May 2010 1/11

Datasheet SHT1x

Sensor Performance

Non functional Parameters

Relative Humidity

Parameter	Condition	min	typ	max	Units
Resolution [1]		0.4	0.05	0.05	%RH
		8	12	12	bit
Accuracy [2] SHT10	typical		±4.5		%RH
	maximal	see Figure 2			
Accuracy [2] SHT11	typical		±3.0		%RH
	maximal	see Figure 2			
Accuracy [2] SHT15	typical		±2.0		%RH
	maximal	see Figure 2			
Repeatability			±0.1		%RH
Hysteresis			±1		%RH
[illegible]	[illegible]		[illegible]		[illegible]
Response time [3]	τ (63%)		8		s
Operating Range		0		100	%RH
Long term drift [4]	normal		±0.5		%RH/yr

Temperature

Parameter	Condition	min	typ	max	Units
Resolution [1]		0.04	0.01	0.01	°C
		12	14	14	bit
Accuracy [2] SHT10	typical		±0.5		°C
	maximal	see Figure 3			
Accuracy [2] SHT11	typical		±0.4		°C
	maximal	see Figure 3			
Accuracy [2] SHT15	typical		±0.3		°C
	maximal	see Figure 3			
Repeatability			±0.1		°C
Operating Range		-40		123.8	°C
		-40		[illegible]	[illegible]
Response Time [6]	τ (63%)	5		30	s
Long term drift			< 0.04		°C/yr

Functional Parameters

Figure 2: Maximal RH-tolerance at 25°C per sensor type.

Figure 3: Maximal T-tolerance per sensor type.

Electrical and General Items

Parameter	Condition	min	typ	max	Units
Source Voltage		2.4	3.3	5.5	V
Power Consumption [5]	sleep		2	5	µW
	measuring		3		mW
	average		90		µW
Communication	digital 2-wire interface, see Communication				
Storage	10 – 50°C (0 – 125°C peak), 20 – 60%RH				

Packaging Information

Sensor Type	Packaging	Quantity	Order Number
SHT10	Tape & Reel	2000	1-100218-04
SHT11	Tape & Reel	100	1-100051-04
	Tape & Reel	400	1-100098-04
	Tape & Reel	2000	1-100524-04
SHT15	Tape & Reel	100	1-100085-04
	Tape & Reel	400	1-100093-04

This datasheet is subject to change and may be amended without prior notice.

[1] The default measurement resolution of is 14bit for temperature and 12bit for humidity. It can be reduced to 12/8bit by command to status register.
[2] Accuracies are tested at Outgoing Quality Control at 25°C (77°F) and 3.3V. Values exclude hysteresis and are applicable to non-condensing environments only.
[3] Time for reaching 63% of a step function, valid at 25°C and 1 m/s airflow.
[4] Value may be higher in environments with high contents of volatile organic compounds. See Section 1.3 of Users Guide.
[5] Values for VDD=3.3V at 25°C, average value at one 12bit measurement per second.
[6] Response time depends on heat capacity of and thermal resistance to sensor substrate.

www.sensirion.com Version 4.3 – May 2010 2/11

Fig. 4.3 A datasheet of a real humidity and temperature sensor SENSIRION SHT1x where functional and nonfunctional attributes are *highlighted*

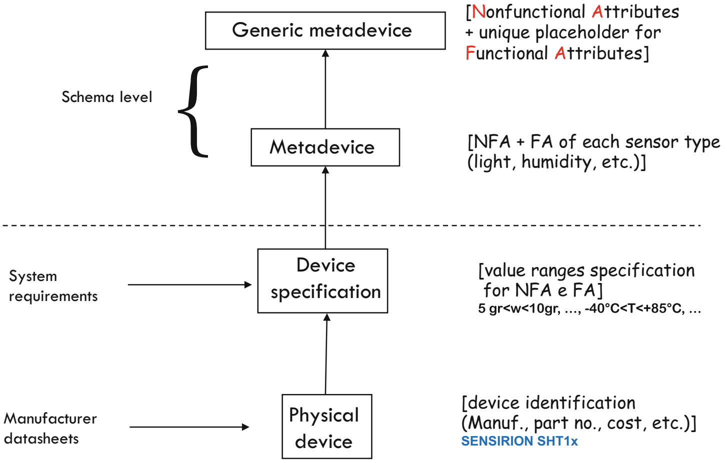


Fig. 4.4 The abstraction process to define the generic metadevice

functional and nonfunctional attributes of a sensor are abstracted from the specific physical sensor (whose description is given in the sensor datasheet) to a generic metadevice. As depicted in Fig. 4.4, the system requirements and the heterogeneity of all the sensor types are taken into consideration in this abstraction process.

This abstraction process can result in the formulation of an extensible markup language (XML) document type definition (DTD) for the generic metadevice that has been defined as follows:

Listing 4.1 An XML document type definition (DTD) for the generic metadevice

```
<!ELEMENT gmd: GenericMetaDevice
 (gmd: weight, gmd: size, gmd: MTBF, gmd: envcond,
 gmd: precision, gmd: metadevice)
 non functional properties of every metadevice-- >
<!ELEMENT gmd: weight (#PCDATA)>
<!ELEMENT gmd: size (gmd: length, gmd: width, gmd: height)>
<!ELEMENT gmd: length (#PCDATA)>
<!ELEMENT gmd: width (#PCDATA)>
<!ELEMENT gmd: height (#PCDATA)>
<!ELEMENT gmd: MTBF (#PCDATA)>
<!ELEMENT gmd: precision (#PCDATA)>
<!ELEMENT gmd: envcond (gmd: tempmax, gmd: tempmin,
 gmd: relhummax, gmd: relhummin)>
<!ELEMENT gmd: tempmax (#PCDATA)>
<!ELEMENT gmd: tempmin (#PCDATA)>
<!ELEMENT gmd: relhummax (#PCDATA)>
<!ELEMENT gmd: relhummin (#PCDATA)>
<!ELEMENT gmd: metadevice (md: light | md: humidity |
 md: temperature | md: waterlevel)

 specific metadevices functional properties -- >
<!ELEMENT md: light (md: lightmax, md: lightmin,
 md wavelenghtmax, md: wavelenghtmin>
<!ELEMENT md: lightmax (#PCDATA)>
<!ELEMENT md: lightmin (#PCDATA)>
<!ELEMENT md: wavelenghtmax (#PCDATA)>
<!ELEMENT md: wavelenghtmin (#PCDATA)>
<!ELEMENT md: humidity (md: relhummax, md: relhummin)>
<!ELEMENT md: relhummax (#PCDATA)>
<!ELEMENT md: relhummin (#PCDATA)>
<!ELEMENT md: temperature (md: tempmax, md: tempmin)>
<!ELEMENT md: tempmax (#PCDATA)>
<!ELEMENT md: tempmin (#PCDATA)>
<!ELEMENT md: waterlevel (md: watlevmax, md: watlevmin)>
<!ELEMENT md: watlevmax (#PCDATA)>
<!ELEMENT md: watlevmin (#PCDATA)>
```

As stated in [9], the word sensor does not necessarily refer only to physical hardware devices (as those detailed in Fig. 4.3) but also to every source of data providing information of interest. Hence, in addition to physical hardware sensors (which is however the most frequently considered source of information in WSNs), virtual and logical sensors are often employed [9]. Virtual sensors refer to nonphysical sources of information; for example, the location of a person can be determined by means of tracking systems through Global Positioning System

(GPS) location (physical devices) as well as by looking at its electronic calendar or by analyzing travel bookings or e-mails (nonphysical sources of information). Electronic calendars, travel bookings, or e-mails do not require a real measurement of the person position and are referred to as virtual sensors. Logical sensors rely on two or more sources of information to provide high-level data; for example, a logical sensor could aggregate data coming from both a physical sensor and a virtual sensor as well as information coming from a database system. Physical, virtual, and logical sensors share the same goal: acquiring information from the application scenario in which they work.

The entities of an application scenario that could be object of the sensing activities can be grouped into three main families [20]:

- *Places*: regions of geographical spaces such as rooms, buildings, locations, or streets
- *People*: individual or groups of people
- *Objects*: physical or artificial objects (e.g., a software application)

Regardless of the kind of sensor (physical, virtual, or logical) or the object of the sensing activities (places, people, or object), [20] defines four essential characteristics or relevant information of the sensing activity:

- *Identity* refers to the ability to univocally identify the entity which is the object of the sensing, e.g., by assigning unique identifiers in the namespace used by the application.
- *Location* encompasses all the aspects related to the position of the sensed entity. It could include not only a position in the space but also orientation, elevation, and proximity to relevant locations.
- *Status* refers to the physical quantity (or the virtual/logical quantity) object of the sensing. In case of a temperature sensor, the status refers to the temperature value measured at a specific time instant.
- *Time* indicates temporal information associated with the sensed measurements. The temporal information often refers to a timestamp specifying the exact time instant at which the measurement has been acquired. A wide range of time coarseness could be considered (e.g., from the nanosecond to the decades) depending on the application needs.

Interestingly, radio-frequency identification (RFID) tags [48] can be considered as a specific type of low-band sensor where the status of the sensor is a code identifier or might refer to more detailed information about a product/object. In particular, passive RFID tags, which do not require battery and are so small to be included in labels, tickets, or articles of clothing, are equipped with an antenna and with a tag identifier (i.e., the status of the sensor). A dedicated device, called RFID reader, powers the passive RFID tag (the antenna of the tag captures the energy) and receives back a signal transmitted from the tag itself (containing the tag identifier). Here, identity and status of the sensing activity are strictly related to the RFID tag and refer to information contained within the tag identifier. Differently, location and time of the sensing activity can be associated to the position and time instant at which the reader interacts with the RFID tag.

4.3 WSN Units: Hardware and Software Description

While sensors represent the technological tool aiming at measuring a physical quantity of interest, WSN units provide the hardware and software framework in which sensors operate and acquired data are processed and transmitted to the data analysis tool. Like sensors, the hardware and software framework of WSN units is characterized by a large heterogeneity in terms of off-the-shelf technological solutions and available operating systems. All these aspects increase the complexity of the designed WSNs and must be taken into account both by the WSN application programmer and by the WSN system designer. Examples of hardware platforms for WSN units are described and commented in Sect. 4.3.1, while Sect. 4.3.2 presents a comparison of the available operating systems for WSNs.

4.3.1 Hardware Platforms

The hardware platforms for WSN units are generally composed of three different modules:

- A processing module
- A transmission module
- A sensing module

The processing module, which is generally equipped with a small memory/storage unit, manages the unit and carries out the processing tasks according to the application needs. The transmission module provides the mechanisms to communicate with the other units of the WSN or to a remote control room. Finally, the sensing module contains one or more sensors, each of which is devoted to measure a physical quantity of interest and generate the associated codeword (i.e., the digital data).

In addition, a power module is generally considered. This module, which is typically application dependent, includes the energy management circuits, the batteries, or energy harvesting solutions (e.g., solar panels).

In the following, some off-the-shelf hardware platforms for WSN units are compared and commented on: CrossBow MICAz,[1] ETH BTnode,[2] EISTEC Mulle,[3]

[1] CrossBow web site—http://www.xbow.com

[2] BTnode web site—http://www.btnode.ethz.ch

[3] EISTEC web site—http://www.eistec.se

Shimmer Research Shimmer,[4] MoteiV TmoteSky [39], CrossBow Imote2, CrossBow StarGate, and NetBrick [8]. The main characteristics of these platforms are summarized in Table 4.2.

As to the processing, the considered hardware platforms encompass different kinds of microprocessors ranging from low-performance (e.g., ATMega128) to high-performance ones (e.g., the PXA series). As expected, a trade-off between performance and power consumption exists (both in sleep and run mode): high-performance microprocessors require larger power consumptions (e.g., 1.45 W for the StarGate), while low-performance ones are characterized by smaller power consumptions (e.g., 60 mW for the MICAz). The choice of a hardware platform is heavily dependent on the task and the role of the unit within the WSN. For example, the MICAz platform is meant for sensing applications requiring low sampling rates and very simple processing, while the StarGate platform is a typical solution for WSN coordinators (units whose goal is to coordinate and manage a set/cluster of units and provide high-level processing abilities). Even the onboard memory reflects the role and the needs of the WSN unit. Hardware platforms meant to act as WSN coordinators are equipped with large external random-access memory (RAM) and flash memory (hence increasing the power consumption of the hardware). Conversely, hardware platforms designed to work as WSN units are only equipped with internal memory embedded within microprocessors (to reduce power consumption). The peculiarity of the NetBrick hardware platform is the ability to act both as a WSN unit (in the configuration with the microprocessor working at 8 MHz and without any external memory chip) and as a WSN coordinator (in the configuration with the microprocessor working at 72 MHz and with large external RAM and flash memories).

As regards the transmission, most of the considered hardware platforms rely on the well-known CC1000 or CC2420 radio modules. These modules are generally characterized by low power consumptions and short transmission ranges (usually between 30 and 100 m). The NetBrick hardware platforms are endowed with the JN5148 radio module, guaranteeing larger transmission ranges than those provided by the CC radio modules at the expense of larger power consumptions. The StarGate is not equipped with a built-in radio module.

As regards the sensing abilities, all the considered hardware platforms are able to host external sensors (through dedicated ADCs). Interestingly, NetBrick, Mulle, Shimmer, and TelosB hardware platforms are directly equipped with onboard sensors (e.g., temperature, humidity, light).

[4] Shimmer web site—http://www.shimmersensing.com

Table 4.2 Some off-the-shelf hardware platforms for WSN units

Platform	Processor					Radio		Sensors
	Type	Frequency	RAM (byte) int./ext.	Flash (byte) int./ext.	Power consumption (mW) sleep/run	Type	TX power (dB)	Types
MICAz	ATMega128	8 MHz	4K/n.a.	128K/n.a.	0.036/60	CC2420	Up to 0	n.a.
BTnode	ATMega128	7.38 MHz	4K/n.a.	128K /n.a.	0.036/60	CC1000	Up to +10	n.a.
Mulle	M16C/62P	Up to 24 MHz	20K/n.a.	128K /n.a.	0.015/75	Bluetooth	Up to +4	Temperature
Shimmer	MSP430	8 MHz	10K/n.a.	48K/µSD	0.006/4	CC2420	Up to 0	MEMS
TelosB	MSP430	8 MHz	10K/n.a.	48K/1 Mb	0.006/4	CC2420	Up to 0	Temperature, humidity, light
Imote2	PXA271	13–416 MHz	256K/32 Mb	32 Mb/n.a.	0.39/ up to 992	CC2420	Up to 0	n.a.
StarGate	PXA255	400 MHz	64K/64 Mb	32 Mb/ext.	15/1455	n.a.	n.a.	n.a.
NetBrick	STM32F103	8–72 MHz	64K/1 Mb	512 Kb/ 128 Mb	0.083/226	JN5148	Up to +20	Temperature, humidity, MEMS

n.a. not available, *ext.* external

Table 4.3 Comparison of operating systems for WSN units

	Architecture	Preemption	Multithreading	Remote reprogramming	Language
TinyOS	Monolithic	No	Partial	No	NesC
Contiki	Modular	Yes	Yes	Yes	C
Mantis	Layered	Yes	Yes	Yes	C

4.3.2 Operating Systems

Operating systems (OSs) represent the intermediate layer between the hardware of the WSN unit and WSN applications. They aim to control the hardware resources and providing services and functionalities to the application software (e.g., hardware abstraction, scheduling of tasks, memory management, and file system). Simplified versions of desktop/mainframe OSs could be considered only in high-performance hardware platforms (e.g., the StarGate platform is able to run a simplified version of Linux). The constraints on hardware and energy of WSN units (as pointed out in Table 4.2) led to the development of OSs specifically devoted to networked embedded systems. Examples of OSs for WSNs are TinyOS [34], Contiki [21], Mantis [10], LiteOS [14], Nano-RK [22], and SOS [25]. In this chapter, we focus on TinyOS, Contiki, and Mantis. A detailed comparison among these three OSs is also presented in Table 4.3.

TinyOS is an open-source, flexible, and component-based OS for WSN units. It is characterized by a very small footprint (400 byte), which makes it particularly suitable for low-power hardware platforms (such as the MICAz). TinyOS relies on a monolithic architecture where software components (including those providing hardware abstraction) are connected together through interfaces. It supports a non-preemptive first-in-first-out scheduling activity (making it unsuitable for computational-intensive or real-time applications). TinyOS applications must be written in NesC (which is a dialect of the C language) and cannot be updated/modified at runtime (applications are compiled together with the OS kernel as a single image). Deluge [29] and Flexcup [38] are software frameworks that allow the remote reprogramming of TinyOS applications on WSN units.

Contiki is an open-source OS for networked embedded systems. It is built around a modular architecture composed of an event-driven-based microkernel, OS libraries, a program loader, and application processes. Interestingly, Contiki supports preemptive multithreading and remote reprogramming (through the program loader). The memory occupation is 2 KB of RAM and 4 KB of read-only memory (ROM). Both the OS and the applications are written in C.

MANTIS is a multithreaded OS for WSNs that is based on a traditional layered architecture. Each layer provides services to upper layers from the hardware to the application threads. MANTIS supports cross-platform design by preserving the programming interface across different platforms. The scheduling activity is based on a preemptive priority-based scheduler, while remote reprogramming is supported by the dynamic loading of threads. Even in this case, both the OS and the application threads are written in C.

4.4 Data Transmissions in WSNs

Once acquired and locally processed at the WSN units, gathered data are then transmitted through the WSN and made available to the information system. Transmissions in WSNs generally rely on a communication stack that is typically composed of three layers [4], i.e., the physical layer, data-link layer, and network layer. More specifically, the physical layer takes care of the low-level communication mechanisms (e.g., signal modulation/demodulation) and depends on the considered transmission medium. The data-link layer [also called media access control (MAC) layer][5] addresses the robust transmission of packets trying to minimize energy consumption and the loss of packets (e.g., due to the collision with neighbors' transmissions). Different mechanisms for the data-link layer could encompass different packet sizes/structures as well as different transmission schemes (e.g., channel content accesses, fixed allocation time slots). Finally, the network layer aims at managing the routing of packets within the network by also considering data aggregation mechanisms and attribute-based or location awareness solutions.

While the first two layers (physical and data link) mainly refer to transmission/communication mechanisms, the data-centric view of WSNs discussed in this chapter led us to deepen the routing solutions for the network layer present in the literature. Interestingly, the constraints on hardware/software, energy, and bandwidth (which are typical of WSNs) as well as the need to manage (possibly) large-scale networks of sensors forced the scientific community to design new and ad hoc routing algorithms for WSNs with different criteria w.r.t. traditional ones. The routing algorithms present in the literature can be grouped into three main families [3, 5]: data-centric, hierarchical, and location-aware routing.

Data-centric protocols rely on a query-based approach where data are requested through queries that are disseminated to the WSN units from the information system. This approach requires attribute-based naming since queries are processed by exploiting the attributes of the physical phenomenon under monitoring. In conjunction with data-centric protocols, data aggregation mechanisms could be considered to reduce the amount of data to be transmitted through the WSN. Examples of data-centric protocols are flooding [26], SPIN (Sensor Protocols for Information via Negotiation) [27], and directed diffusion [30]. Flooding relies on a simple mechanism where each WSN unit receiving a packet broadcasts it to all its neighborhoods up to when the packet reaches its destination. The main drawback of this protocol is the fact that it generates duplicate messages sent to the same WSN

[5] AA.VV. The ieee 802.15.4 standard—http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf, 2006.

unit. SPIN (Sensor Protocols for Information via Negotiation), which is one of the earliest solutions for data-centric routing, relies on high-level descriptors (metadata) for attribute-based naming. Metadata are exchanged among WSN units before transmissions. Once a WSN unit receives a packet, the neighbors are advertised, and those who are interested in the contained data request the packet through an ad hoc message. This solution allows for reducing the amount of messages that are transmitted through the WSN with respect to the flooding algorithm. Directed diffusion, which is one of the most important data-centric protocols present in the literature, is based on the idea that data are diffused through the WSN units by relying on a naming scheme. More specifically, direct diffusion relies on attribute-value pairs for the data, and WSN units are queried in an on-demand manner through these pairs. Each unit can employ aggregation and caching mechanisms, and the query-based mechanism of directed diffusion is highly energy efficient and does not require a knowledge of the global topology of the network.

Hierarchical protocols focus on the scalability of WSNs. In order to reduce latency and energy consumption of single-tier wireless networks, hierarchical protocols rely both on network clustering and on data aggregation/fusion mechanisms. Clusters of WSN units are generally created according to units' proximity and residual energy in batteries. LEACH (Low-Energy Adaptive Clustering Hierarchy) [28] is one of the most popular hierarchical protocols for WSNs. Clusters are formed according to signal strength, and cluster heads (i.e., a WSN unit whose duty is to coordinate the other units of the cluster) act as routers of the clusters of units. Other examples of hierarchical protocols are TEEN [37], LLC [31], and XLLC [7].

Finally, location-aware protocols exploit the positions of WSN units to route data through the network in an energy-efficient way. For instance, following this approach, a query could be diffused only to those WSN units present in a specific location (encompassing both fixed and mobile WSN units). Distances between WSN units could be estimated by analyzing incoming signal strength, while the location could be made available through low-power GPS modules. For example, the Geographic Adaptive Fidelity algorithm [50] suggests dividing the network area into zones, and WSN units are associated with a zone according to their geographical position. WSN units belonging to a zone collaborate and cooperate for routing and application purposes. Differently, the Geographic and Energy Aware Routing algorithm [52] proposes the use of heuristics based on energetic and location information to select the neighbors to which a packet must be routed. Other examples of location-aware routing algorithms are MECN/SMECN [42], GOAFR [33], and SPAN [16].

4.5 Querying a WSN

While the previous sections presented the technological, software, and algorithmic solutions of WSNs, this section focuses on the management of data in WSN applications. To this purpose, we emphasize that the large heterogeneity of the pervasive systems, comprising devices ranging from passive RFID tags to application servers, requires a very high level of transparency so that as many technical problems as possible can be handled by the system, while the application programmer only writes high-level codes. Moreover, the application programmer should also be relieved from knowing the physical structure of the network and how to address single devices.

As described in Sect. 4.2, data produced by a WSN are characterized by peculiar characteristics that must be taken into account by the application programmer in the design of WSN applications [46]:

- Data are produced as continuous streams by sensors, with no need for an explicit request, and often at well-defined time intervals.
- Each value is associated with a timestamp.
- Data represent real-world phenomena that often require real-time processing.
- The high volume and rate of data in a data stream make it difficult to "park" it on a disk storage for further processing, so in-network data processing procedures are often required, at least in the first steps.

Some more comments are in order:

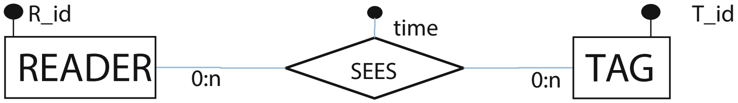
- Not all of the sensor readings are of interest to the applications, so that they can often be summarized and possibly some of them can even be discarded.
- Data from sensors can be affected by noise and uncertainty or can be missing.
- Power is the most precious resource for a sensor; therefore, energy-aware optimization procedures must be used in every step of data management.

These considerations have driven a number of efforts to define and implement high-level languages for managing data in WSN applications, privileging the "what to do" with respect to the "how to do" approach and preferring declarative languages to procedural ones; such a choice also allows an easier in-network distributed data processing managed at the middleware level.

As an example of the database view and of declarative queries on sensors, in Fig. 4.5 a simple entity-relationship conceptual model of an RFID system, such as the one presented in Sect. 4.2, is shown together with an instance of the corresponding relational table.

Let us suppose now that, by entering a large archaeological area (e.g., the Pompeii site), each visitor is endowed with a bracelet containing an RFID and that readers can be located at the most interesting points of the site. The site manager could be interested in monitoring the visit paths followed by visitors or in studying how long visitors stand by a point of interest (e.g., inside the Casa dei Vettii). The first case amounts to considering RFID tags as (virtual) short data stream generators (one per visitor), while in the second case, RFID readers are the data stream generators (one per interest point) and the RFID tags are the data. In listing 4.2, the SQL code for answering these queries is shown.

Another example of the database view of a WSN is shown in Fig. 4.6, where the entity-relationship conceptual model for a WSN-based system is presented



rfidsys

R_id	T_id	time
R1	T15	t1
R1	T54	t5
R1	T15	t7
R2	T22	t2
R2	T15	t3
R2	T81	t4
R3	T15	t5
R3	T22	t6

Fig. 4.5 Conceptual model and a relational instance of an RFID system

Listing 4.2 SQL queries for archaeological site monitoring

```
QUERY 1: Which Readers  Tag T15 passed by in the last ten ↩
    minutes?

SELECT  R_id
FROM  rfidsys
WHERE  T_id=T15 AND time IN
    (SELECT(time,T_id)
    FROM  rfidsys
    WHERE  (t-time) < 10)

QUERY 2: Which Tags passed under Reader R1 in the last ten ↩
    minutes?

SELECT  T_id
FROM  rfidsys
WHERE  R_id=R1 AND time IN
    (SELECT(time,T_id)
    FROM  rfidsys
    WHERE  (t-time) < 10)
```

for monitoring the risks—floods, fire, earthquake, and others—incurred by the "objects" on the site (e.g., monuments, frescos, etc.).

The most frequent query mode in sensor networks is continuous query operation; queries are downloaded to the peripheral devices together with some time- or event-based measurement conditions, and data are collected accordingly with no need for

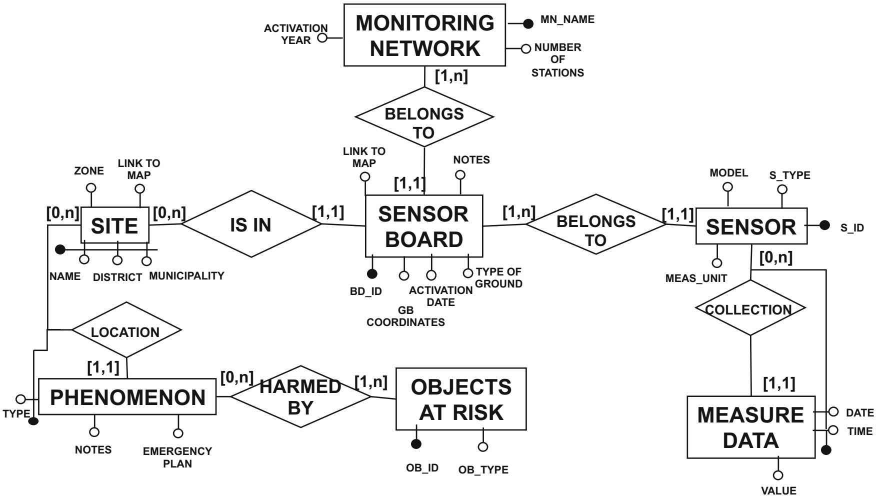


Fig. 4.6 Conceptual model of a WSN-based environmental risk monitoring system

further polling actions. In [36], Madden et al. clearly state the critical issues to be faced for processing a query in a WSN: (1) finding out the sensors that have data relevant to the query, (2) when sampling for answering the query should occur, (3) in which order the samples should be taken and how to schedule sampling with respect to other operations, and (4) evaluating the trade-offs between the lower energy-consuming local computations and the higher energy required to send data to a collecting station.

Therefore, two phases can be distinguished: (1) the dissemination of queries to the sensing nodes, which requires choosing effective routing mechanisms (from simple flooding mechanisms to more complex routing algorithms), and (2) the collection of the sensed data, in which the minimum amount of data required by the application must be sent back. It must be noted that the requirement that the activation of the energy-consuming radio link must be kept at minimum makes query processing and optimization in WSNs different from what is usually done in DBMSs. We already said that not all the data sampled by sensors are necessary or even useful to the applications; therefore, data compression, aggregation, or shredding can be used on the nodes in order to reduce the data volume to be transmitted [13, 32, 35]. Dynamically setting data collection frequencies is particularly useful in WSN in order to optimize power-consuming transmission operations while limiting storage occupation. On the other hand, the quality of data (QoD) is often of paramount importance if we consider that sensors can be used in safety critical or sensitive monitoring processes. The QoD can be defined by three measures [15]:

- *Accuracy*, usually defined as the degree of conformity of a measured or computed quantity to its actual (true) value

- *Precision*, that is, the degree to which further measurement or calculations show the same or similar results
- *Timeliness*, defined as the property of information to arrive early or at the right time

While accuracy and precision can be affected by compression/aggregation techniques, another issue to be considered in query optimization is the fact that data are transferred to the base station using some standard transmission protocol (typically at the MAC level) which, as described in Sect. 4.4, usually uses packets as the unit of transmission. Therefore, the packet dimension and the transmission delays must be considered: owing to the fixed length of header and trailer, larger data packets are more energy efficient; moreover, the usage of slotted protocols eliminates the risk of message collision. However, the fixed packet length can make the benefits of compression/aggregation vanish, and the time needed to assemble large packets or to wait for the proper transmission slot can badly affect data timeliness. Therefore, query optimization in WSNs must result from a balanced blend of energy sparing, QoD, and transmission protocol choices that derive from the interplay among the MAC level, routing, and query processing [47, 51].

4.6 WSN Data Languages

In the following, we briefly introduce some of the most noticeable approaches to query processing in a WSN, among which are TinyDB, global sensor networks (GSN), and declarative sensor networks (DSN); then we describe PerLa, a language for data management in pervasive and context-aware systems. A rich tutorial and survey on how to program a WSN can be found in [40].

4.6.1 TinyDB

TinyDB [36] is one of the first projects that introduced the idea of abstracting a WSN as a database. In this way, the final user can define the desired data gathering by writing an SQL-like query. At the higher level, an interface is provided to inject queries in the system. Each query is sent from the user workstation to the WSN base station; then, the query is distributed to all the nodes, and data are acquired from the sensors in the environment and filtered, possibly aggregated, and routed out to the base station. Multiple persistent queries with different sampling times are issued from an information system. TinyDB is defined over TinyOS, and it exploits a power-efficient in-network processing algorithm. The main constraint of TinyDB is that only homogeneous WSNs composed of TinyOS-based devices can be managed.

4.6.2 GSN

GSN [1, 2] is a scalable, lightweight system that can be easily adapted, even at runtime, to new types of sensors, thus allowing a dynamic reconfiguration of the system. The GSN middleware is based on the concept of virtual sensor, a software component able to filter and process data coming from the wrappers—software components that transform raw data from a physical source into a GSN-suitable format—or from other virtual sensors. The behavior of these components is defined through an XML configuration file, while the required data processing is set by using an SQL-like query. Unfortunately, virtual sensors are to be developed specifically for every new kind of node. A remarkable feature of GSN is the support for heterogeneous devices which, however, is achieved by requiring support for Java programming at the device level or, at least, the availability of a TCP/IP stack; these requirements can be an obstacle for the deployment on devices characterized by reduced computational capabilities and memory availability.

4.6.3 DSN

Even if, in principle, the DSN project [17, 18] is similar in scope to TinyDB, it uses a completely different approach. The whole system is specified and implemented by means of the declarative language Snlog—a dialect of Datalog; Snlog is used at all levels - from data acquisition to network and transmission management - but for the device drivers, which are written in NesC and can be invoked through a built-in predicate mechanism. The authors claim that their approach allows programmers the flexibility to choose their own ratio of declarative to imperative code, and they also acknowledge that "DSN is not adapt at natively manipulating opaque data objects such as timeseries, matrices and bitmasks; nor is it fit for providing real-time guarantees" [17].

4.6.4 PerLa

PerLa (PERvasive LAnguage)[6] is an SQL-like language designed to allow for querying a pervasive system as if it were a database. The language is supported by a middleware which masks the idiosyncrasies of the nodes employed in complex sensing networks and performs the runtime management of the system [43, 44].

The goal of the language is to enable the final user to collect data in a fast and easy way without dealing with low-level programming issues. The idea behind

[6] The PerLa web site—http://perlawsn.sourceforge.net/index.php

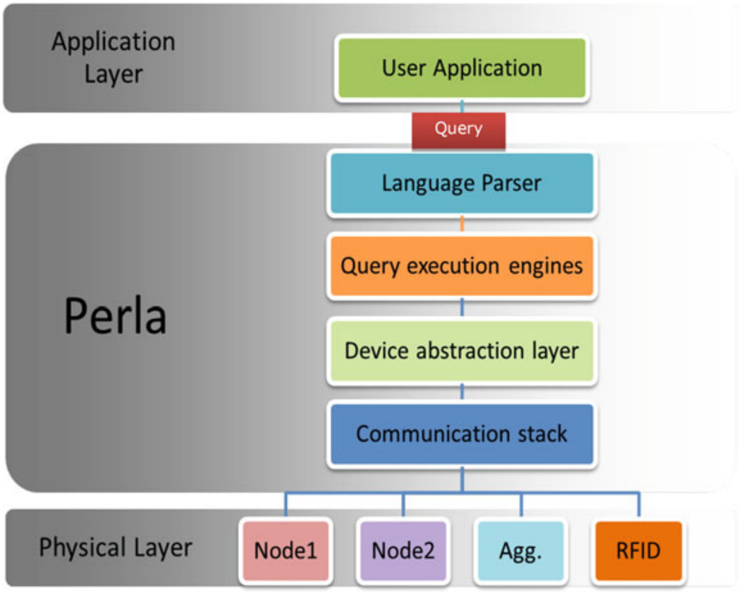


Fig. 4.7 Gross architecture of the PerLa system

PerLa is to extend the database-like abstraction from a single WSN unit to a whole pervasive system and to provide full support for heterogeneity, both at runtime and at deployment time.

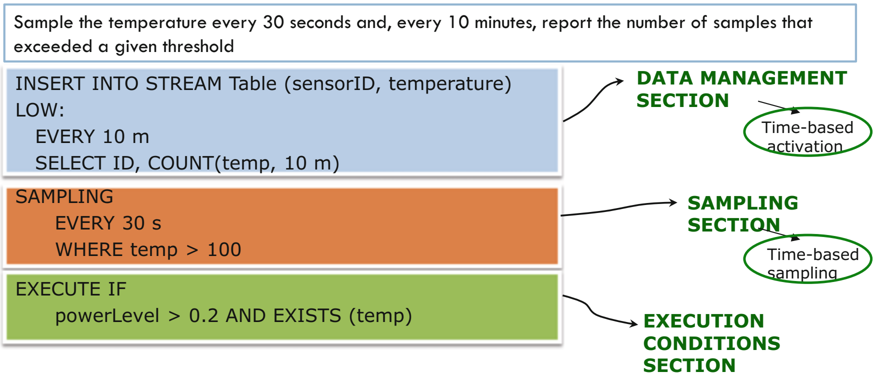
The language allows the user to interact with logical objects, called functionality proxy component (FPC), which wrap (and mask) physical devices. An FPC can abstract both a single node and a set of devices (e.g., a whole WSN); FPCs have common and homogeneous interfaces and are used by PerLa queries to access the data gathered from the network nodes. The FPC is defined as a Java object representing a physical device; it must be instantiated on a system capable of (1) running a Java virtual machine (JVM) and (2) connecting itself to a TCP/IP network. Knowledge of the nodes' hardware and computational characteristics is not needed to perform a PerLa query. Moreover, by means of the FPC abstraction, the language is not tied to any particular type of sensing device, including social networks and messaging systems. Figure 4.7 shows the gross architecture of the system.

PerLa support for pervasive systems extends to node developers as well. The addition of new sensing devices in an existing network is facilitated by a plug and play connection system, i.e., a runtime factory that generates all the software components needed to query new sensor nodes. The information required to automatically assemble a device driver is stored in an XML file provided by the node developer. This file—the device descriptor—details all the nodes' characteristics in terms of data structures, protocols of communication, computational capabilities, and behavioral patterns. The descriptor can be sent by the device itself upon start-up or directly injected by the user (e.g., for RFIDs or other dumb devices). All running queries will automatically make use of every new node added in the system. A huge work toward standardizing information about sensors and actuator is being carried on by the Open Geospatial Consortium (OGC), which defined the SensorML model and XML Encoding Standard [12].

Collecting data from a pervasive system abstracted by PerLa can be as easy as writing a typical database query. Moreover, PerLa is composed of special syntactic statements designed to fully exploit the capabilities typical of pervasive sensing networks [43, 44].

Two types of data structures are supported: (1) *STREAM TABLES*, which are unbounded lists of records on which queries can perform the insertion of new records and extract a data window with parameters [$timestamp, size$], and (2) *SNAPSHOT TABLES*, which store the set of records produced by a query in a given period T and the content of which is refreshed every period T. These data structures are manipulated by three kinds of queries:

- *Low-level queries (LLQ)* are the most original part of the language. They allow to precisely define the behavior of a single device. The main role of an LLQ is to define the sampling operations but also to allow the application of some SQL operators (grouping, aggregation, filtering) on sampled data. When an LLQ is injected into the system, the set of logical objects that meet all the requirements to execute is computed. Then, an instance of the query is deployed on each selected logical object: data produced by all these instances are collected in a stream, which can be further manipulated by high-level queries or directly sent as output to the final user. In order to provide the needed expressiveness for managing the interaction with logical objects (e.g., definition of the sampling parameters), LLQ present some relevant syntactical differences with respect to standard SQL [43]. In Fig. 4.8, the typical structure of an LLQ is shown. The *Sampling Section* specifies how and when the sampling operation should be performed; sampling is simply defined as the generation of a record, whose fields are filled with the current values of the logical object attributes. The *Data Management Section* has the role of managing sampled data to compute the query results. The syntax of this query section is similar to the standard SQL one, but some important differences exist due to the need of managing, through the definition of windows, the ideally infinite buffer (the local buffer), where all the sampled records are appended. The *Execution Conditions Section* defines the rules to establish if a certain logical object should participate to the query. This section is optional, and, if not specified, all the logical objects in the system will be involved in the query execution.
- *High-level queries (HLQ)* allow to define data manipulation over the streams generated by other LLQ or HLQ. These kinds of queries do not directly deal with the logical object abstraction since they operate only on streams, independently from their sources. As a consequence no ad hoc clauses are needed: the syntax and the semantics of these statements are similar to those of streaming DMSs.



Sample the temperature every 30 seconds and, every 10 minutes, report the number of samples that exceeded a given threshold

```
INSERT INTO STREAM Table (sensorID, temperature)
LOW:
  EVERY 10 m
  SELECT ID, COUNT(temp, 10 m)
SAMPLING
    EVERY 30 s
    WHERE temp > 100
EXECUTE IF
    powerLevel > 0.2 AND EXISTS (temp)
```

Fig. 4.8 Typical structure of a low-level query

- *Actuation queries (AQ)* add the capability to modify the state of logical objects, enabling in this way a complete interaction with the environment. These kinds of statements are often employed to control a physical actuator or to set some parameters used by low-level algorithms executed directly on the device (e.g., the sampling frequencies or the thresholds to generate events).

  In summary, the goals of the PerLa middleware are:

  – Provide an abstraction for each device
  – Avoid the introduction of a specific clause for each nonfunctional feature, but rather to provide metadata management exactly alike to sampled data management
  – Support the execution of PerLa queries
  – Make the definition and the addition of new devices (and new technologies) easy, reducing the amount of the needed low-level code by means of a plug and play support which allows devices to automatically start executing queries when they are powered on

4.6.4.1 PerLa Evolution

Born as a simple WSN query language, the functionality of PerLa has evolved toward a true pervasive system language. As shown in [19], the first step has been to extend the language syntax and semantics in order to allow context definition, starting from a context dimension tree (CDT) context model. Furthermore, the middleware has been extended to allow context-aware data management, and work is being carried on to include also calls to layers of a context-oriented programming language and to web services [41].

4.7 Summary

The main goal of this chapter was to introduce the basic components of a pervasive system—sensors, nodes, and networks—showing how they can be uniformly abstracted by means of conceptual models and how middleware and languages can be designed in order to manage them with a declarative “database style.” We think that this approach can be very useful for allowing application designers to focus on the data they need without bothering with the very different features of the devices.

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: VLDB ‘06: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1199–1202. VLDB Endowment (2006)
2. Aberer, K., Hauswirth, M., Salehi, A.: The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical Report. LSIR-REPORT- 2006-006, pp. 1–21 (2006)
3. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad Hoc Netw. **3**(3), 325–349 (2005)
4. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
5. Al-Karaki, J.N., Kamal, A.E.: Routing techniques in wireless sensor networks: a survey. IEEE Wirel. Commun. **11**(6), 6–28 (2004)
6. Alippi, C.: Intelligence for Embedded Systems A Methodological Approach. Springer, Berlin (2014)
7. Alippi, C., Camplani, R., Roveri, M.: An adaptive llc-based and hierarchical power-aware routing algorithm. IEEE Trans. Instrum. Meas. **58**(9), 3347–3357 (2009)
8. Alippi, C., Camplani, R., Roveri, M., Viscardi, G.: Netbrick: a high-performance, low-power hardware platform for wireless and hybrid sensor networks. In: IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS), 2012, pp. 111–117. IEEE, Las Vegas, USA (2012)
9. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
10. Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., Han, R.: Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms. Mobile Netw. Appl. **10**(4), 563–579 (2005)
11. Bonnet, P., Gehrke, J., Seshadri, P.: Toward sensor database systems. In: Proceedings of the 2nd International Conference on Mobile Data Management (MDM01). Lecture Notes in Computer Science, pp. 3–14. Springer, Berlin (2001)
12. Botts, M., Robin, A.: Ogc sensorml: model and xml encoding standard, v.2. Open Geospatial Consortium, pp. 1–196 (2014)
13. Boulis, A., Ganeriwal, S., Srivastava, M.B.: Aggregation in sensor networks: an energy accuracy trade-off. Ad Hoc Netw. **1**(23), 317–331 (2003). doi:http://www.dx.doi.org/10.1016/S1570-8705(03)00009-X. http://www.sciencedirect.com/science/article/pii/S157087050300009X. Sensor Network Protocols and Applications
14. Cao, Q., Abdelzaher, T., Stankovic, J., He, T.: The liteos operating system: towards unix-like abstractions for wireless sensor networks. In: International Conference on Information Processing in Sensor Networks, 2008, IPSN’08, pp. 233–244. IEEE, St. Louis, USA (2008)

15. Cappiello, C., Schreiber, F.A.: Experiments and analysis of quality and energy-aware data aggregation approaches in wsns. In: QDB 2012 10th International Workshop on Quality in Databases (Co-Located with VLDB 2012), pp. 1–8 (2012)
16. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. Wirel. Netw. **8**(5), 481–494 (2002)
17. Chu, D., Tavakoli, A., Popa, L., Hellerstein, J.: Entirely declarative sensor network systems. In: VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1203–1206. VLDB Endowment (2006)
18. Chu, D., Popa, L., Tavakoli, A., Hellerstein, J.M., Levis, P., Shenker, S., Stoica, I.: The design and implementation of a declarative sensor network system. In: SenSys '07: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, pp. 175–188. ACM, New York, NY (2007)
19. Colace, F., Moscato, V., Quintarelli, E., Rabosio, E., Tanca, L.: Context awareness in pervasive information management. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems, Springer, Heidelberg/New York (2015)
20. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum. Comput. Interact. **16**(2), 97–166 (2001)
21. Dunkels, A., Gronvall, B., Voigt, T.: Contiki-a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks, 2004, pp. 455–462. IEEE, Tampa, USA (2004)
22. Eswaran, A., Rowe, A., Rajkumar, R.: Nano-rk: an energy-aware resource-centric rtos for sensor networks. In: 26th IEEE International Conference on Real-Time Systems Symposium, 2005. RTSS 2005, 10 pp. IEEE, Miami, USA (2005)
23. Fowler, K.R., Schmalzel, J.L.: Sensors: the first stage in the measurement chain. IEEE Instrum. Meas. Mag. **7**(3), 60–65 (2004)
24. Fowler, K.R., Schmalzel, J.: Why do we care about measurement? IEEE Instrum. Meas. Mag. **7**(1), 38–46 (2004)
25. Han, C.C., Kumar, R., Shea, R., Kohler, E., Srivastava, M.: A dynamic operating system for sensor nodes. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, pp. 163–176. ACM, Seattle, USA (2005)
26. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18**(4), 319–349 (1988)
27. Heinzelman, W.R., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 174–185. ACM, Seattle, USA (1999)
28. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000, 10 pp. IEEE, Hawaii, USA (2000)
29. Hui, J.W., Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 81–94. ACM, Baltimore, USA (2004)
30. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 56–67. ACM, Boston, USA (2000)
31. Kim, J., Kim, S., Kim, D., Lee, W., Kim, E.: Low-energy localized clustering: an adaptive cluster radius configuration scheme for topology control in wireless sensor networks. In: 2005 IEEE 61st Vehicular Technology Conference, 2005. VTC 2005-Spring, vol. 4, pp. 2546–2550. IEEE, Stockholm, Sweden (2005)
32. Kimura, N., Latifi, S.: A survey on data compression in wireless sensor networks. In: ITCC (2), pp. 8–13 (2005)

33. Kuhn, F., Wattenhofer, R., Zollinger, A.: Worst-case optimal and average-case efficient geometric ad-hoc routing. In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 267–278. ACM, Annapolis, USA (2003)
34. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al.: Tinyos: an operating system for sensor networks. In: Weber, W., Rabaey, J.M., Aarts, E. (eds.) Ambient Intelligence, pp. 115–148. Springer, Berlin (2005)
35. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. **36**(SI), 131–146 (2002). doi:10.1145/844128.844142. http://www.doi.acm.org/10.1145/844128.844142
36. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1), 122–173 (2005). http://www.doi.acm.org/10.1145/1061318.1061322
37. Manjeshwar, A., Agrawal, D.P.: Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In: IPDPS, vol. 1, p. 189 (2001)
38. Marrón, P.J., Gauger, M., Lachenmann, A., Minder, D., Saukh, O., Rothermel, K.: Flexcup: a flexible and efficient code update mechanism for sensor networks. In: Römer, K., Karl, H., Mattern, F. (eds.) Wireless Sensor Networks, pp. 212–227. Springer, Berlin (2006)
39. TMote Sky Datasheet http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
40. Mottola, L., Picco, G.P.: Programming wireless sensor networks: fundamental concepts and state of the art. ACM Comput. Surv. **43**(3), 51 (2011). http://www.doi.acm.org/10.1145/1922649.1922656
41. Panigati, E., Schreiber, F.A.: Context-aware software approaches: a comparison and an integration proposal (discussion paper). In: Proceedings of the 22nd Italian Symposium on Advanced database Systems, pp. 175–184 (2014)
42. Rodoplu, V., Meng, T.H.: Minimum energy mobile wireless networks. IEEE J. Sel. Areas Commun. **17**(8), 1333–1344 (1999)
43. Schreiber, F.A., Camplani, R., Fortunato, M., Marelli, M.: Design of a declarative data language for pervasive systems. Art Deco Technical Report R. A. 11.1b. http://perlawsn.sourceforge.net/documentation.php?official=1 (2008)
44. Schreiber, F.A., Camplani, R., Fortunato, M., Marelli, M., Rota, G.: Perla: A language and middleware architecture for data management and integration in pervasive information systems. IEEE Trans. Softw. Eng. **38**(2), 478–496 (2012). doi:10.1109/TSE.2011.25
45. Schreiber, F.A., Tanca, L., Camplani, R., Viganó, D.: Pushing context-awareness down to the core: more flexibility for the perla language. In: Proceedings of the 6th PersDB 2012 Workshop (Co-Located with VLDB 2012), pp. 1–6 (2012)
46. Schreiber, F.A., Panigati, E., Zaniolo, C.: Data streams and data stream management systems and languages. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Heidelberg/New York (2015)
47. Sharaf, M.A., Beaver, J., Labrinidis, A., Chrysanthis, P.K.: Balancing energy efficiency and quality of aggregate data in sensor networks. VLDB J. **13**(4), 384–403 (2004)
48. Want, R.: An introduction to rfid technology. IEEE Pervasive Comput. **5**(1), 25–33 (2006). doi:10.1109/MPRV.2006.2
49. Weiser, M.: The computer for the 21st century. Sci. Am. **265**(3), 66–75 (1991)
50. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad hoc routing. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 70–84. ACM, Rome, Italy (2001)
51. Yates, D.J., Nahum, E.M., Kurose, J.F., Shenoy, P.J.: Data quality and query cost in pervasive sensing systems. Pervasive Mob. Comput. **4**(6), 851–870 (2008)
52. Yu, Y., Govindan, R., Estrin, D.: Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report, Technical report ucla/csd-tr-01-0023, UCLA Computer Science Department (2001)

Chapter 5
Data Streams and Data Stream Management Systems and Languages

Emanuele Panigati, Fabio A. Schreiber, and Carlo Zaniolo

5.1 A Short Introduction to Information Flow Processing and Data Streams

The massive usage of data streams dates back to artificial satellite information processing systems and to their commercial application in the early 1970s, such as in telecommunications switching, land monitoring, meteorological surveillance, etc. Today they are extensively used in monitoring systems applications based on wired and wireless sensor networks, in social networks, and in the Internet of Things [20]. The main functional goals of data stream management systems (DSMSs) are as follows: (a) results must be pushed to the output promptly and eagerly while input tuples continue to arrive and (b) because of the unbounded and massive nature of data streams, all past tuples cannot be memorized for future use. Only synopses can be kept in memory and the rest must be discarded. The main problem created by these goals is a further significant loss of expressive power with respect to the well-known limitations of structured query language (SQL) and other relational languages; unfortunately, these limitations are dramatically more disruptive to data stream applications for the following reasons: (1) blocking query operators that were widely used on databases can no longer be allowed on data streams; (2) database libraries of external functions written in procedural languages are much less effective in DSMSs, which process data in small increments rather than as aggregate large objects; and (3) embedding queries in a procedural programming

E. Panigati • F.A. Schreiber (✉)
DEIB – Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milan, Italy
e-mail: emanuele.panigati@polimi.it; fabio.schreiber@polimi.it

C. Zaniolo
UCLA Computer Science Department, 3532D Boelter Hall, Los Angeles, CA 90095-1596, USA
e-mail: zaniolo@cs.ucla.edu

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_5

language, a solution used extensively in relational database applications, is of very limited effectiveness on data streams. A key research issue for DSMSs is deciding which data model and query language should be used; a wide spectrum of different solutions have in fact been proposed, including operator-based graphical interfaces, programming language extensions, and an assortment of other solutions provided in publish/subscribe systems. In this chapter, the main features and architectural issues will be examined for DSMSs as an introduction to the following chapters, which provide a detailed account of their usage in different application areas. In particular, we shall introduce several *stream query languages* and *data stream management systems* (DSMSs) developed in different projects and contexts.

5.2 Data Stream Definitions

In this section, we will provide some definitions and general concepts about data streams and their handling, allowing the reader to completely and easily understand the peculiarity of each of the languages and systems presented in Sect. 5.4. Without loss of generality, we shall mostly refer to relational data streams, i.e., streams whose data elements are constituted by relational tuples over a schema $\mathfrak{R}$.

Table 5.1 summarizes some of the most noticeable differences between database management systems (DBMSs) and DSMSs as far as their functional features are concerned.

First of all, let us discuss some peculiar differences between a DBMS and a DSMS. A sort of antisymmetry can be noticed between the data management functions in the two of them: in the database, data are stored in a permanent way, while queries are dynamically created and processed; however, in data streams, queries are permanently stored in the sensors and continuously produce a data stream flowing from the real world to a final processing/storage station. Therefore, DSMS computation is generally *data driven*, computing the results as new data come into the system (or at least periodically), since queries are *continuous queries*,

Table 5.1 Comparison between DBMS and DSMS functional features

Feature	DBMS	DSMS
Model	*Persistent* data	*Transient* data
Table	*Set or bag* of tuples	*Infinite sequence* of tuples
Updates	All	Append only
Queries	Transient	Persistent
Query answers	Exact	Often approximate
Query evaluation	Multi-pass	One pass
Operators	Blocking and non-blocking	Non-blocking
Query plan	Fixed	Adaptive
Data processing	Synchronous	Asynchronous
Concurrency overhead	High	Low

which are issued only once and permanently remain active in the system; the DBMS approach is *query driven*, and computation runs when a specific query is executed (one-time queries). Therefore, the first problem a DSMS must deal with is managing changes in data arrival rate during a specific query lifetime.

Another problem is related to the unboundedness of the stream itself that continuously flows into the system and cannot be stored in a traditional way; therefore, only the most recent data should be used for answering the queries at a given time point.

In order to make these concepts more precise, let us introduce some formal definitions:

- A *stream* S is a countably infinite set of elements $s \in S$.
- A *stream element* is a four-tuple s: $< v, t^{app}, t^{sys}, bid >$ where
  - $v \in \Re$ is a relational tuple
  - $t^{app} \in T$ is a partially ordered application time value in the time domain T
  - $t^{sys} \in T$ is a totally ordered system time value in T
  - $bid \in \mathbf{N}$ is a batch-id value, where a *batch* B is a finite subset of S where all $b \in S$ have an identical t^{app}

A fundamental concept in data stream systems is that of *window*: due to the unboundedness of the stream, it is necessary to define some slices on it—called windows—in order to allow correct processing in finite time. The window *size* can be defined as a fixed number of data items—also called *snapshots*—or as a fixed time interval, no matter how many data items are included in it. In the latter case, also the *slide*, i.e., the time distance between two consecutive windows, and the possibility of windows overlapping, as in the case of sliding windows shown in Fig. 5.1, must be considered.

A *time-based window* W *=(o,c]* over S is a finite subset of stream elements s where $o < s.t^{app} \leq c$:

- *Window size* ω: $\forall\ W = (o, c] \in W \Rightarrow (c - o) = \omega$
- *Window slide* β: $\exists\ W_1, W_2 \Rightarrow (o_2 - o_1) = \beta$

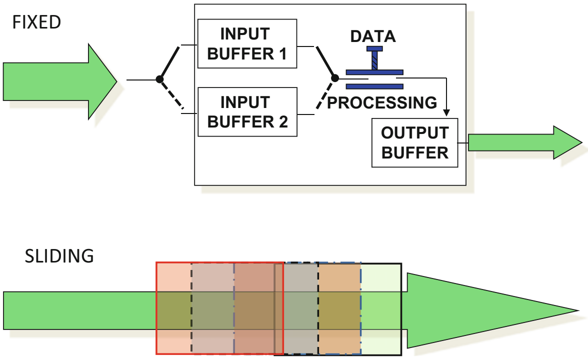


Fig. 5.1 Example of fixed and sliding windows

If $\beta \geq \omega$, windows are disjoint; if $\beta < \omega$, windows are (partially) overlapped.

More definitions and some examples on the usage of windows in the continuous query language (CQL) [3] will be introduced in Chap. 7.

Systems and applications that handle data streams must allow to filter the stream, to compute aggregates, to compose several streams in a unique stream (or decompose a unique stream to several streams), and to find frequent data in the stream and perform some online mining on it.

A stream could be processed in two main ways, *forward* and *backward*, depending on the processing engine implementation, the query language semantics, and the initial processing time point. Both these processing paradigms present problems in handling infinite streams:

Backward: The system may have started long before the moment at which a given query/rule processing has to be evaluated; therefore, lots of data should be processed, causing a heavy workload and very long response time. The solution to the backward infinity-looking problem is to set a backward-looking limit, thus avoiding the system from searching for solutions from very old data and focusing only on the top, most recent ones.

Forward: *Monotonic* operators, such as COUNT and SUM, are *non-blocking* since they can work on data as soon as they arrive; however, some operators, called *blocking operators* (e.g., the `NOT` operator), require that a single query/rule has to process a finite set of data which must be known when the computation starts; otherwise their execution may be delayed infinitely, causing processes to starve. The solution chosen by the community for this problem is to use *time windows*. When the computation starts, the operator considers only a time-limited subset of the stream, allowing it to give a timely answer to the query/rule.

Moreover, as described in [35], data stream processing engines, in order to perform real-time processing, must ensure compliance with the following eight basic rules:

Keep the data moving. Messages should be processed online, without the need to be stored in any kind of repository before the analysis takes place. The processing model should be active and should not involve any kind of source polling.

Query using stream-enabled SQL. The chosen query language should provide stream-oriented semantic operators and primitives to allow effective querying of the stream.

Handle stream imperfections. The DSMS must be able to handle without issues any kind of imperfection in the stream (e.g., delayed, missing, or out-of-order data) giving the queries consistent answers. Real-world systems have plenty of these issues, making this a crucial feature of the system.

Generate predictable output. The system must ensure that equivalent data sequences in the stream produce the same output results.

Integrate stored and streaming data. A DSMS should allow the seamless integration among streaming and static data (e.g., data stored in a DBMS) possibly allowing to query them adopting the same language. Moreover, the system should also be able to keep track of its internal state, allowing it to be resumed if necessary.

Guarantee data safety and availability. DSMSs have to ensure high availability since they often operate in critical conditions. Furthermore, they must ensure safe behavior in case of failures.

Automatic partition and scale. The system should be able to transparently distribute its workload among multiple—and different—machines and processors, improving its scalability.

Real-time processing. The engine must be implemented in a highly optimized way introducing a minimal overhead in the processing operation, producing the queries' results in real-time and also in high-data-rate contexts.

The last requirement, together with the consideration that incomplete data can often be tolerated, led to the definition of *load shedding* [5, 37]; this technique is used when the input rate from the data sources exceeds the system capacity, thus causing a system overload and an increase of output latency. In order to fulfill quality-of-service (QoS) constraints and real-time deadlines, some data items (tuples) are discarded. In [37], several techniques are presented for inserting and removing tuple *drop* operators from query plans, either in a random way or following the semantics of the application. The authors discuss *when* and *where* load should be shed and *how much* load to shed from the query plan while maintaining response quality degradation within acceptable limits.

5.3 Data Stream Structure and Constraints on Queries

A stream can assume several different structures, the original definition of stream "an append-only, ordered sequence of timestamped items" [21], having been relaxed in many ways. For instance, if the concept of *revision tuple* is introduced in the language, some tuples are not appended to the stream, but they simply replace other tuples [29].

If the stream is not continuous, but is burst based, it could be viewed as a sequence of sets [38] or, in another possible representation, as a sequence of relational tuples [3] or objects [9, 36].

According to the presence or absence of the timestamp among the attributes of the tuple and to the kind of preprocessing applied before the tuple reaches the system, four different models of stream systems can be defined [19]:

Unordered cash register. Unordered items come from different domains without any kind of preprocessing.

Ordered cash register. Time-ordered items come from different domains without any kind of preprocessing.

Unordered aggregate. Items coming from the same domains are preprocessed and only one aggregate item per domain is appended to the stream without following any particular order.

Ordered aggregate. Items coming from the same domains are preprocessed, and only one aggregate item per domain is appended to the stream following some time-based order.

From the queries' point of view, the property of *monotonicity* ensures that the result of a query can be updated incrementally as data flow in the system; i.e., if $Q(t)$ is the answer to the query Q at time t, executing the query at two different time instants t_i and t_j, $Q(t_i) \subseteq Q(t_j)$, $\forall t_j > t_i$.

Moreover, for monotonic queries, the following property holds:

$Q(t) = \bigcup_{t_i=1}^{t} (Q(t_i) - Q(t_{i-1})) \cup Q(0)$

The property simply states that it is enough to reevaluate the query considering only newly arrived tuples in the stream, appending the new results to the former ones.

On the other hand, *non-monotonic* query results cease to be valid once new items—involved in the query computation—are appended to the stream. Examples of non-monotonic queries are queries that involve *blocking* operators like the NOT (negation) operator that cannot return its answer until all the data to be considered have flown into the system.

5.4 Data Stream Query Languages and Data Stream Management Systems

The main issue encountered while trying to apply the traditional DBMS model to data stream processing is related to the necessity of building a persistent storage where all the relevant data is kept and whose updates are relatively rare (almost true for traditional relational databases but not for data streams). This approach has a negative trend on performance if there are many rules to be processed (more than a given threshold) or when the data arrival rate is too high.

To overcome these limitations, data stream management systems (DSMSs)—a new class of systems oriented toward processing large streams of data in a timely way—have been developed by the database community in different projects and contexts during the last years, having different semantics associated to the queries. The answer to a query can be seen as an append-only output stream or as an entry in a storage that is continuously modified as new elements flow inside the processing stream. An answer can be either exact or approximate, depending on the system memory that has been (and can be) used to store all the required elements of input streams' history [5, 37].

As stated in [16], a DSMS can be modeled as a set of standing queries Q, one or more input streams, and four possible outputs:

The *Stream* is composed of all the elements of the answer that are produced once and never changed.

The *Store* contains the parts of the answer that may be changed or removed at a certain future time point. The stream and the store together define the current answer to Q.

The *Scratch* represents the working memory of the system, where it is possible to store data useful to compute the answer but that is not part of the answer.

The *Throw* is a sort of recycle bin that is used to throw away unneeded tuples.

The model described above is the most complete to define the behavior of DSMSs, showing how the DSMS's only approach cannot entirely cover the needs for IFP (information flow processing), introducing the need of some other kinds of paradigm for complex pattern detection and notification [16, 26].

In the following, a brief presentation of some of those languages and systems is given.

5.4.1 StreaQuel and TelegraphCQ

TelegraphCQ [11] is a general-purpose continuous query system based on a declarative, SQL-based language called *StreaQuel.*

StreaQuel is a relational, SQL-derived stream query language. StreaQuel manages unbounded flows by means of its native window operator *WindowIs*. Several WindowIs operators may be used in a query, one for each input window, embedded in a *for* loop, which is part of each rule and defines a time variable indicating when the rule has to be processed.

The general assumption behind this mechanism is that it must consider an absolute time model. By adopting an explicit time variable, StreaQuel enables users to define their own policy for moving windows. As a consequence, each window may contain a different number of elements, since its dimension is not bounded a priori.

TelegraphCQ is based on a C++ implementation as an extension of PostgreSQL. The compilation of rules into query plans is made through adaptive modules [4, 27, 32], which dynamically decide how to route data to operators and how to order commutative operators. As explained in [33], thanks to this modularity, TelegraphCQ can be spread on multiple machines with a clustered distribution approach, taking into account processing constraints only.

5.4.2 XML-QL and NiagaraCQ

NiagaraCQ [12] is an IFP system for Internet databases, providing a high-level abstraction to retrieve information from a frequently changing environment like Internet sites. NiagaraCQ is based on the *XML-QL* (extensible markup language-query language) [18].

XML-QL is a declarative, relationally complete query language for extensible markup language (XML); its simple structure allows to easily extend well-known relational database query optimization techniques extracting data from existing XML documents and creating new ones.

The initial draft of this language is presented in a W3C note.[1] In [18], the authors present an extension of XML that allows to manage *ordered* XML documents. This extension introduces in the query language concepts like *tag variables*, *regular path expressions*, multiple-source integration, and *user-defined* functions that allow advanced XML processing.

While the NiagaraCQ engine is designed to run in a totally centralized way, information sources are actually distributed. To increase scalability, NiagaraCQ uses an efficient caching algorithm, which splits operators into groups, keeping members of the same group in the same query plan, thus increasing performance.

5.4.3 OpenCQ

Like NiagaraCQ, *OpenCQ* [25] is an IFP system developed to deal with Internet-scale update monitoring. In OpenCQ, rules are divided into three parts:

- The definition of the operations to be performed on data (in SQL)
- The activation trigger condition
- The stop condition

OpenCQ supports the same processing and interaction models as NiagaraCQ.

OpenCQ has been implemented on top of the DIOM framework [24]: a client-server communication paradigm is used to collect rules and information and to distribute results, keeping the rule processing completely centralized. Wrappers allowing integration among different and heterogeneous sources are used in order to transform the input flow to the required results output flow. Each wrapper can behave in a *pull* or in a *push* way, depending on the source behavior: in the first scenario, the wrapper periodically retrieves data from the source, while in the second, the wrapper simply waits for the source to send in new data that it needs to prepare for processing.

5.4.4 Tribeca

Tribeca [36] has been developed for network traffic monitoring and fits this particular domain. Its rules take a single information flow as input and produce one or more output flows. Rules are expressed using an imperative language, defining the sequence of operators through which the information must flow.

[1] http://www.w3.org/TR/NOTE-xml-ql/

Three different operators exist in the language: selection (called qualification), projection, and aggregate; it is also possible to split (merge) flows by means of ad hoc multiplexing (demultiplexing) operators. Tribeca supports both count-based and time-based windows, which can be sliding or tumble. Windows are used to specify the part of input flows to be processed when performing aggregates. Since items do not have an explicit timestamp, the processing is performed in arrival order. Since the windows are time based, it is impossible to count the number of information items captured at each iteration. Moreover, a rule may be satisfied by multiple and different sets of elements (multiple selection policy), while an element may be part of the computation in more than one iteration, as there is no explicit consumption policy. Rules are translated into direct acyclic graphs in order to be executed, applying graph optimization techniques during the transformation in order to improve performance.

5.4.5 CQL and Stream

Stream computes a query plan starting from *CQL* rules (CQL, *continuous query language*, is an SQL-Like declarative language developed by the database group of Stanford University in the same project), taking into account predefined performance policies in order to schedule the execution of the operators in the plan.

The *transforming rule* is the basic concept of *CQL* [3]; it provides a unified syntax for processing both information flows and stored relations.

CQL provides a clear semantic for the rules, together with a simple language to define them. It defines operators that manage the data stream: due to the fact that a stream is a potentially infinite bag of timestamped data items, these operators extract finite bags of data. The CQL operators split into three different classes as shown in Fig. 5.2:

Relation-to-relation operators derived directly from the SQL operators are the basic, core operators of the language and allow the definition of rules using an SQL-like standard notation (e.g., relational algebraic expressions).

Stream-to-relation operators transform streams in relations; the most well-known operator of this class is the *window* operator that supports flow processing by defining *sliding*, *pane*, and *tumble* windows on the stream.

Relation-to-stream operators (*IStream*, *DStream*, and *RStream*) define how to generate new information flow starting from tuples.

CQL uses a multiple selection policy, associated with a zero consumption policy (the tuples in the stream are never consumed; they still exist after their processing is completed).

Stream provides two shedding techniques to deal with the resource overload problem that is typical of data streams:

- The first one applies load shedding on a set of sliding window aggregation queries to reduce the impact on limited computational resources [5].

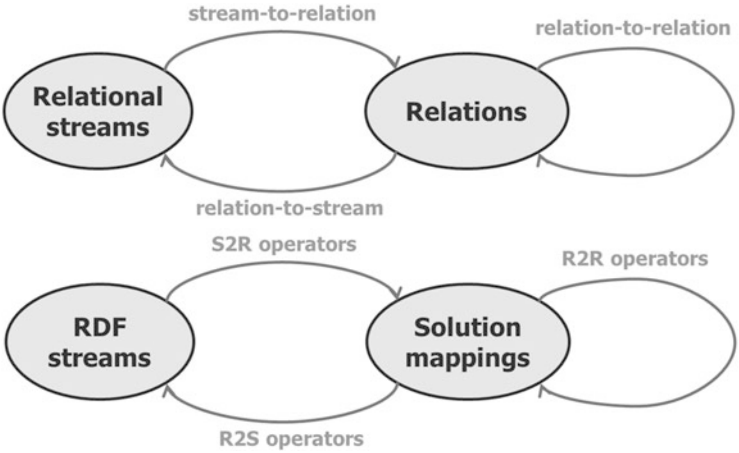


Fig. 5.2 The CQL model

- The second one discards the state of the operator for a set of windowed joins, reducing the impact on memory occupation (in case of limited memory) [34].

The CQL model inspired the design of different Resource Description Framework (RDF) stream processing engines [7, 10, 23], and currently there are different implementations (e.g., C-SPARQL, SPARQLstream, CQELS) of the adapted model (see Chap. 7 [17]). The stream and the relation concepts are mapped to RDF streams and to a set of mappings (using the SPARQL algebra terminology[2]), respectively. To highlight the similarity of RSP operators to those of CQL, similar names are used: S2R, R2R, and R2S to indicate the operators analogous to stream-to-relation, relation-to-relation, and relation-to-stream operators. Chapter 7 [17] presents a description of (1) the sliding window as an example of S2R operators, (2) SPARQL algebra as an example of R2R operators, and (3) the streaming operator as an instance of R2S operators.

5.4.5.1 SQuAl and Aurora/Borealis

Aurora [1] is a general-purpose DSMS that defines transforming rules using an imperative language called *SQuAl*.

SQuAl defines rules in a graphical way, by means of boxes and arrows, connecting different operators of the rule.

SQuAl operates on windows, processing either a single tuple at a time or several tuples at a time, applying a specific user-defined function. This allows information tuples to be used more than once (multiple selection policy), while there exists no consumption policy.

[2] Cf. http://www.w3.org/TR/sparql11-query/

SQuAl allows multiple input processing and multiple result outputs, allowing to restrict the output by means of ad hoc QoS policies in order to fit users and/or application requirements. SQuAl defines the necessary load-shedding policies and customizes the system behavior according to QoS constraints.

Aurora enforces QoS constraints to automatically define shedding policies. Input and output flows do not have an associated timestamp but are processed in their arrival order.

One of the most interesting features of Aurora is the possibility to include intermediate storage points in the rule plan as needed to keep historical information and to recover after operators' failure.

The processing is organized by a scheduler, taking an optimized version of the user-defined plan and choosing how to allocate computational resources to different operators according to their load and to the specified QoS constraints.

Some project extensions have been created in order to investigate distributed processing inside both a single administrative domain and over multiple domains [13]. The goal of these extensions, called *Aurora** and *Medusa*, is that of efficiently distributing the load between available resources; in these implementations, processors communicate using an overlay network, with dynamic bindings between operators and flows.

Recently, the two projects have been merged into the Borealis stream processor [2], which includes all the two previous project features.

5.4.6 GSQL and Gigascope

Gigascope [14, 15] is a DSMS specifically designed for network applications, including traffic analysis, intrusion detection, performance monitoring, etc. Its main concern is to provide high performance for the specific application field it has been designed for. Gigascope is based on a declarative, SQL-like language, called GSQL.

Gigascope translates GSQL rules into basic operators, composing them into a processing plan and using optimization techniques to rearrange the plan according to the nature and the cost of each operator.

GSQL is a declarative, SQL-like language, which includes only filters, joins, group by, and aggregates.

The processing paradigm used by GSQL is very different from other data stream engines. To manage blocking operators, instead of introducing the usual concept of window, it assumes that every input tuple has at least one ordered (monotonically increasing or decreasing) attribute (e.g., the timestamp), using this attribute as a constraint in the processing. For example, the join operator, by definition, must have a constraint on an ordered attribute for each involved stream.

This paradigm has a limited set of application domains (the ones in which it is possible to make strong assumptions on the nature of data and on their arrival order), but it makes the processing semantics easier to understand and similar to that of traditional SQL.

5.4.7 PerLa

The PerLa middleware is a collection of software units designed to support the execution of PerLa queries. Its design revolves around the Functionality Proxy Component (FPC), a proxy object which acts as a decoupling element between sensing nodes and middleware users. In Chap. 4 [30], PerLa is introduced as a pervasive data management language; in this chapter, we briefly introduce the structure of its middleware which, reduced to its essence, constitutes the software environment needed to manage the life cycle of a set of FPCs.

PerLa's middleware provides an easy, XML-based mechanism which allows for a fast integration of new sensors in the system; the designer must only provide a sensor description by means of a standard XML syntax. When the system detects the new sensor descriptor, the parser automatically produces the software implementation of the sensor, and the sensor will be instantly integrated, dispatching its own sensed data as required by the queries.

Being a database-like language, in PerLa the queries' results are automatically stored in a relational database to provide a history of data. In its current development stage, PerLa bases itself on an existing DBMS.

Figure 5.3 shows a high-level snapshot of the whole PerLa system architecture. There are two main runtime layers in the system: a *low-level support* layer, a *hardware abstraction layer*, and the central body of the PerLa middleware itself, which provides standard mechanisms and APIs to handle all the different devices that compose the pervasive system; and a *high-level support* layer that provides query execution services and data management.

Figure 5.4 shows in more detail what the low tier of the middleware is composed and how it operates in order to generate FPCs, while Fig. 5.5 shows the internal structure of a single FPC. Further details about the PerLa middleware are available in [28].

PerLa uses the Channel Manager to create a channel virtualization that allows establishing a bidirectional point-to-point channel between the FPC and the device; this channel can simulate plug and play connections, recognizing new devices when inserted automatically. FPCs are created by the FPC factory, which receives the XML file containing the details about the device and creates an FPC adapter for the node. The FPC registry, on the other hand, maintains the list of all the known devices and their relative FPC.

The PerLa's user interface is constituted of the Query Parser that receives the queries directly from the user (to whom the execution results are provided); after having verified their syntax and validated their semantics, it sends them to the Query Analyzer. The Query Analyzer is responsible for the creation of the low-level query (LLQ) executor that manages the communication with the physical layer and the high-level query (HLQ) executor that is implemented as a wrapper for external data management systems and permits the user to perform data management operations. A LLQ executor is created for each FPC involved, configured for the execution of the statement submitted. FPCs are then consulted for getting data from the physical devices, and results are pushed all together to an output stream.

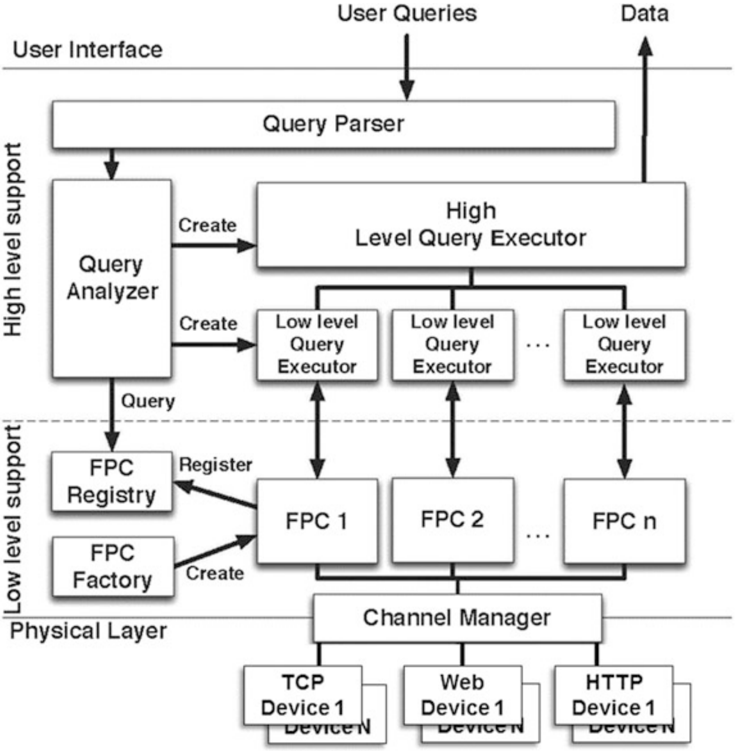


Fig. 5.3 Schema of PerLa's architecture

When nodes are connected to a shared broadcast bus—as in real applications—the channel must be multiplexed: an Adapter Server is added to the middleware to perform (de)multiplexing operations on the FPC side, while on the other side of the connection, an Adapter Client manages the communication channel [8].

The *PerLa query language* [31] is an extension of standard SQL, which includes clauses that allow defining the scope of each of the three kinds of queries:

- HLQ, *high-level queries* which aggregate information coming from different sensors
- LLQ, *low-level queries* which run directly on the sensors to set sampling frequencies and to state how to add data to the stream, possibly performing some kind of aggregation or thresholding on the sensor itself
- AQ, *actuation queries* whose content is a set of commands that activate devices that have capabilities of performing some kind of action (e.g., a small electric engine or a valve)

Further details about the PerLa query language have been presented in Chap. 4 [30].

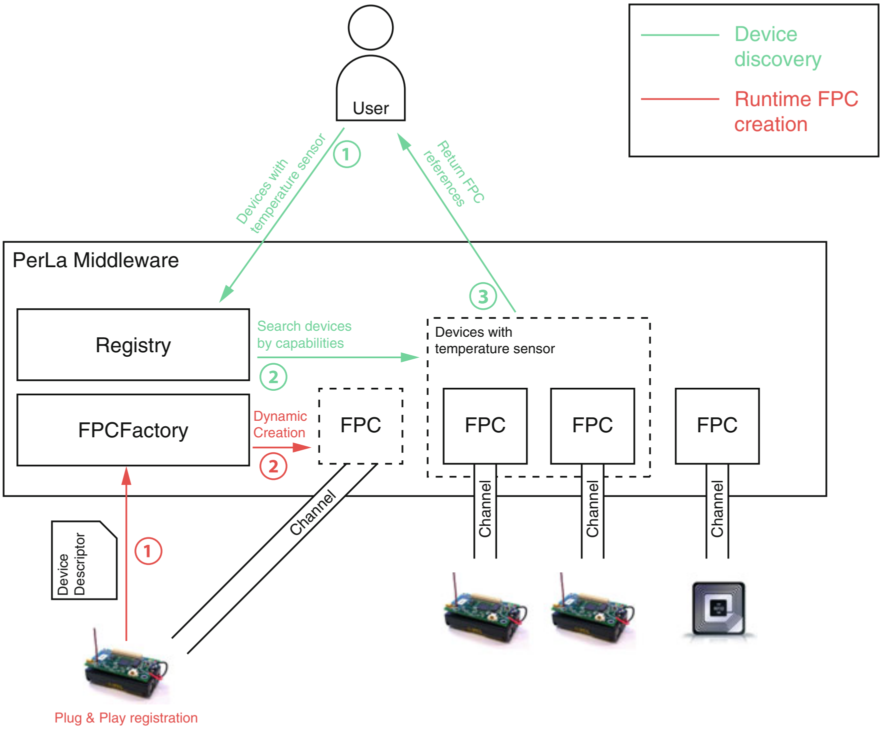


Fig. 5.4 PerLa middleware architecture

5.5 The Stream Mill System

Stream Mill [6] is a general-purpose DSMS that supports an SQL-like continuous query language called Expressive Stream Language (ESL). ESL provides new constructs whereby users can define complex tasks such as data mining tasks and the flow of information between query operators. Therefore, ESL is a highly expressive language which extends the declarative query constructs of SQL with the ability of creating new “user-defined aggregates” (UDAs). UDAs are defined by declaring internal tables and a triplet of statement blocks preceded by the keyword INITIALIZE, ITERATE, and TERMINATE.

The blocks contain update statements that operate on the internal tables. Thus, the statements in INITIALIZE define the operations performed at the beginning of the stream (i.e., when the first tuple is processed), whereas the ITERATE statements define the processing associated with the remaining tuples and TERMINATE specifies the closing computation to be performed after the last tuple. Thus, to define the traditional average aggregate, the user will declare an internal table where the pair $(V, 1)$ is stored by statements in the INITIALIZE block as soon as the first value V of the stream is detected; then, in the ITERATE block, the current pair of values

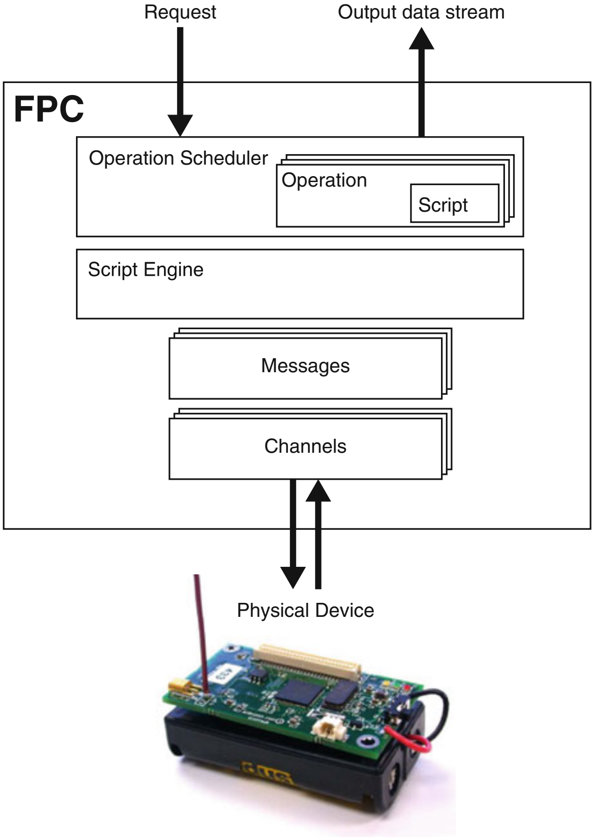


Fig. 5.5 FPC internal architecture

is increased by $(+V, +1)$ for each new arriving input value V. Finally, when the EOF is detected, the statements in the TERMINATE block will compute the ratio between the sum and the count accumulated in the internal table and return it as the final result of the aggregate.

UDAs turn SQL into a Turing-complete language [22] that can express data mining algorithms concisely [39]. However, the UDA just described and every UDA that uses TERMINATE to return its final results are blocking and thus cannot be used on data streams.

A first solution to this problem consists of returning all results in ITERATE, as in the case of cumulative aggregates that produce a new result for each new incoming tuple. As a second solution, SQL-2 aggregates and blocking UDAs can be called on windows whereby, for each new input tuple, they return their value on the whole window—a value that for most aggregates can be computed efficiently from the tuples that just expired out of the window. Thus, ESL allows an additional group of UDA statements to be declared under the EXPIRE label to manage this differential computation.

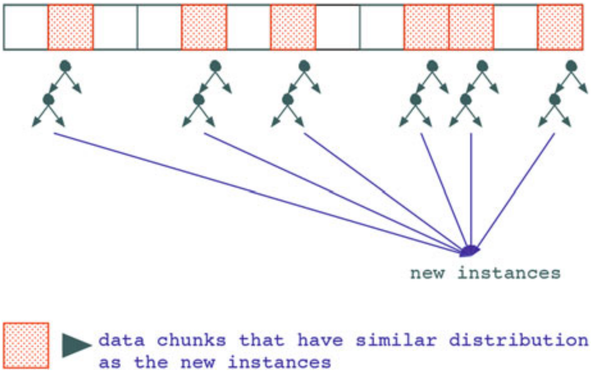

new instances
data chunks that have similar distribution as the new instances

Fig. 5.6 Mining streams with concept drift

These constructs turn ESL into a powerful language for managing a wide range of continuous queries on data streams efficiently, including the search for complex patterns, sketch aggregates, exponential histograms, and approximate frequent item sets [40]. Load shedding techniques that speed up processing by obtaining approximate results can also be expressed through UDAs, which entail the application of popular data mining algorithms to data streams [40].

In fact, using constructs such as window panes, it is often possible to mine a stream considering each chunk of the stream by itself, deriving a model from it and then combining the models as shown in Fig. 5.6 [6]. This approach can also be used to reduce the classification error in environments prone to concept drift, which occurs when the statistical properties of the data changes over time in unpredictable ways lowering the accuracy of the predicted model.

Stream Mill keeps the traditional architecture of a database system; it compiles rules into a query plan whose operators are then dynamically scheduled. The implementation of Stream Mill as described in [6] shows a centralized processing, where the engine exchanges data with applications using a client-server communication paradigm.

Figure 5.7 shows the overall system architecture.

In particular, the system is based on a multiple-client, single-server architecture, allowing the user to define her/his own streams, queries, and aggregates using the client query editor which also displays the processing result to the user. On the server side, the system parses and compiles the continuous queries, generating a query graph that describes which of the tuples have to be extracted from the input buffer and then how they are to be processed in order to be sent to the output buffer. Any UDA is eventually translated in *C*++ language, in order to be as fast and efficient as native ESL language operators.

Buffer management policies are adopted to avoid useless waste of main memory, removing tuples from the input buffers as they are processed (except from tuples involved in window-based computation).

Stream Mill also allows the use of different execution policies depending on the user needs: sometimes it is necessary to receive results in real time, so the system

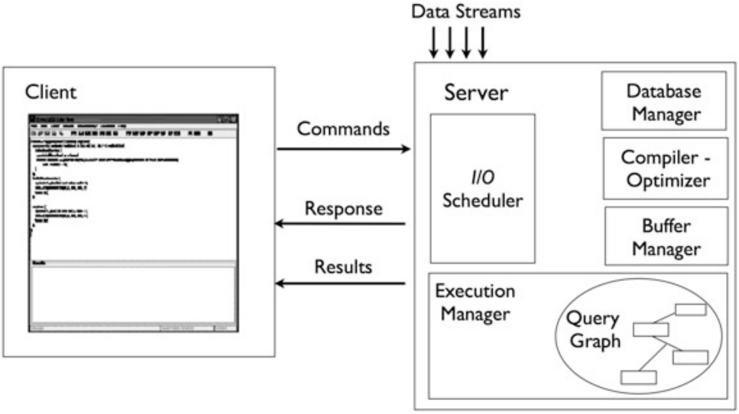


Fig. 5.7 The Stream Mill system architecture

adopts a policy to reduce the response time, while in low-memory scenarios a policy to reduce the main memory occupation will fit best, reducing the necessary memory space to store partial computation results. The switching between different policies is performed at runtime, in a quick and efficient way, without stopping the system query processing.

5.6 Summary and Conclusions

In this chapter, we have introduced some basic concepts on data streams and data stream processing and have reviewed some of the available DSMSs.

In the last section of this chapter, we have also presented in detail *Stream Mill*, a DSMS based on an SQL-like query language providing interesting stream processing features that allow the mining of streams.

The concept presented in this chapter will be the ground on which many of the systems and results presented in the following chapters are built.

References

1. Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E., Hatoun, M., Maskey, A., Rasin, A., et al.: Aurora: a data stream management system. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, p. 666. ACM, New York (2003)
2. Abadi, D.J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., et al.: The design of the borealis stream processing engine. In: CIDR, vol. 5, pp. 277–289 (2005)

3. Arasu, A., Babu, S., Widom, J.: The cql continuous query language: semantic foundations and query execution. J. Int. J. Very Large Data Bases **15**(2), 121–142 (2006)
4. Avnur, R., Hellerstein, J.M.: Eddies: continuously adaptive query processing. In: ACM SIGMoD Record, vol. 29, pp. 261–272. ACM, New York (2000)
5. Babcock, B., Datar, M., Motwani, R.: Load shedding for aggregation queries over data streams. In: Proceedings of 20th International Conference on Data Engineering, 2004, pp. 350–361. IEEE, Boston (2004)
6. Bai, Y., Thakkar, H., Wang, H., Luo, C., Zaniolo, C.: A data stream language and system designed for power and extensibility. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 337–346. ACM, New York (2006)
7. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying rdf streams with c-sparql. SIGMOD Rec. **39**(1), 20–26 (2010). doi:10.1145/1860702.1860705. http://doi.acm.org/10.1145/1860702.1860705
8. Bolchini, C., Orsi, G., Quintarelli, E., Schreiber, F.A., Tanca, L.: Progettazione dei dati con l'uso del contesto. Mondo Digitale (2008)
9. Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. In: Mobile Data Management, pp. 3–14. Springer, Heidelberg (2001)
10. Calbimonte, J.P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I, ISWC'10, pp. 96–111. Springer, Heidelberg (2010). http://dl.acm.org/citation.cfm?id=1940281.1940289
11. Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S.R., Reiss, F., Shah, M.A.: Telegraphcq: continuous dataflow processing. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 668–668. ACM, New York (2003)
12. Chen, J., DeWitt, D.J., Tian, F., Wang, Y.: Niagaracq: a scalable continuous query system for internet databases. In: ACM SIGMOD Record, vol. 29, pp. 379–390. ACM, New York (2000)
13. Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., Zdonik, S.B.: Scalable distributed stream processing. In: CIDR, vol. 3, pp. 257–268 (2003)
14. Cranor, C., Gao, Y., Johnson, T., Shkapenyuk, V., Spatscheck, O.: Gigascope: high performance network monitoring with an sql interface. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 623–623. ACM, New York (2002)
15. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: a stream database for network applications. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 647–651. ACM, New York (2003)
16. Cugola, G., Margara, A.: Processing flows of information: from data stream to complex event processing. ACM Comput. Surv. **44**(3), 15 (2012)
17. Dell'Aglio, D., Balduini, M., Della Valle, E.: Applying semantic interoperability principles to data stream management. In: Data Management in Pervasive Systems, Chapter 7. Springer, Berlin (2015)
18. Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Suciu, D.: A query language for xml. Comput. Netw. **31**(11), 1155–1169 (1999)
19. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Surfing wavelets on streams: one-pass summaries for approximate aggregate queries. In: VLDB, vol. 1, pp. 79–88 (2001)
20. Golab, L., Özsu, M.T.: Issues in data stream management. ACM Sigmod Rec. **32**(2), 5–14 (2003)
21. Guha, S., McGregor, A.: Approximate quantiles and the order of the stream. In: Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 273–279. ACM, New York (2006)
22. Law, Y., Wang, H., Zaniolo, C.: Relational languages and data models for continuous queries on sequences and data streams. ACM Trans. Database Syst. **36**(2), 8 (2011). doi:10.1145/1966385.1966386. http://doi.acm.org/10.1145/1966385.1966386
23. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Proceedings of the 10th International

Conference on the Semantic Web - Volume Part I, ISWC'11, pp. 370–388. Springer, Berlin (2011). http://dl.acm.org/citation.cfm?id=2063016.2063041

24. Liu, L., Pu, C.: A dynamic query scheduling framework for distributed and evolving information systems. In: Proceedings of the 17th International Conference on Distributed Computing Systems, 1997, pp. 474–481. IEEE, Baltimore (1997)
25. Liu, L., Pu, C., Tang, W.: Continual queries for internet scale event-driven information delivery. IEEE Trans. Knowl. Data Eng. **11**(4), 610–628 (1999)
26. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Springer, Berlin (2011)
27. Raman, V., Raman, B., Hellerstein, J.M.: Online dynamic reordering for interactive data processing. In: VLDB, vol. 99, pp. 709–720 (1999)
28. Rota, G.: Design and development of an asynchronous data access middleware for Pervasive Networks: the case of PerLa. Master's thesis, Politecnico di Milano (2014)
29. Ryvkina, E., Maskey, A.S., Cherniack, M., Zdonik, S.: Revision processing in a stream processing engine: a high-level design. In: Proceedings of the 22nd International Conference on Data Engineering, 2006. ICDE'06, pp. 141–141. IEEE, Washington (2006)
30. Schreiber, F.A., Roveri, M.: Sensors and wireless sensor networks as data sources: models and languages. In: Data Management in Pervasive Systems, Chapter 4. Springer, Berlin (2015)
31. Schreiber, F.A., Camplani, R., Fortunato, M., Marelli, M., Rota, G.: Perla: a language and middleware architecture for data management and integration in pervasive information systems. IEEE Trans. Softw. Eng. **38**(2), 478–496 (2012). doi:10.1109/TSE.2011.25
32. Shah, M.A., Hellerstein, J.M., Chandrasekaran, S., Franklin, M.J.: Flux: an adaptive partitioning operator for continuous query systems. In: Proceedings of 19th International Conference on Data Engineering, 2003, pp. 25–36. IEEE, Los Alamitos (2003)
33. Shah, M.A., Hellerstein, J.M., Brewer, E.: Highly available, fault-tolerant, parallel dataflows. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 827–838. ACM, New York (2004)
34. Srivastava, U., Widom, J.: Memory-limited execution of windowed stream joins. In: Proceedings of the 30th International Conference on Very Large Data Bases, vol. 30, pp. 324–335. VLDB Endowment, Toronto (2004)
35. Stonebraker, M., Çetintemel, U., Zdonik, S.: The 8 requirements of real-time stream processing. ACM SIGMOD Rec. **34**(4), 42–47 (2005)
36. Sullivan, M., Heybey, A.: Tribeca: a system for managing large databases of network traffic. In: Proceedings of USENIX (1998)
37. Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Load shedding in a data stream manager. In: Proceedings of the 29th International Conference on Very large Data Bases, vol. 29, pp. 309–320. VLDB Endowment, Berlin (2003)
38. Tucker, P.A., Maier, D., Sheard, T., Fegaras, L.: Exploiting punctuation semantics in continuous data streams. IEEE Trans. Knowl. Data Eng. **15**(3), 555–568 (2003)
39. Wang, H., Zaniolo, C.: Atlas: A native extension of SQL for data mining. In: D. Barbará, C. Kamath (eds.) Proceedings of the 3rd SIAM International Conference on Data Mining, San Francisco, 1–3 May 2003, pp. 130–141. SIAM, San Francisco (2003). doi:10.1137/1.9781611972733.12. http://dx.doi.org/10.1137/1.9781611972733.12
40. Zaniolo, C.: Mining databases and data streams with query languages and rules. In: F. Bonchi, J. Boulicaut (eds.) Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID 2005, Porto, 3 October 2005. Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 3933, pp. 24–37. Springer, Berlin (2005). doi:10.1007/11733492_2. http://dx.doi.org/10.1007/11733492_2

Chapter 6
The Complex Event Processing Paradigm

Gianpaolo Cugola and Alessandro Margara

6.1 Introduction

As mentioned in the previous chapters, pervasive systems require continuous processing of information collected from multiple sources deployed in the environment under analysis. Often, a large portion of such information encodes *notifications of events* that occurred within the pervasive system or in the environment where it operates. In such cases, the goal of the processing step is to detect *situations of interest* as soon as they occur by looking at the *primitive events* that have been observed.

On the one hand, this requires the ability to define situations of interest (often called *composite events*) as *patterns* of primitive events, joined by specific relationships. On the other hand, this also demands the capability of effectively exploiting such definitions to detect composite events at runtime, as soon as they occur. *Complex event processing* (CEP) languages and systems have been proposed by research and industry to satisfy these complementary needs.

The CEP paradigm shares some similarities with the data streaming approach described in Chap. 5. Nevertheless, several key differences exist. Specifically, data stream management systems (DSMSs) focus on *transforming* the incoming streams of information into new streams, e.g., by joining or aggregating data items. Conversely, CEP systems interpret incoming information as notifications of events and focus on *detecting* relevant patterns among such streaming notifications. This

G. Cugola (✉)
Dipartimento di Elettronica, Politecnico di Milano, Informazione e Bioingegneria (DEIB),
Piazza L. da Vinci 32, Milan, Italy
e-mail: gianpaolo.cugola@polimi.it

A. Margara
Faculty of Informatics, Università della Svizzera Italiana, Via Buffi 13, Lugano, Switzerland
e-mail: alessandro.margara@usi.ch

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric
Systems and Applications, DOI 10.1007/978-3-319-20062-0_6

chapter focuses on this specific processing abstraction, describing the key features and issues that characterize it.

We discuss CEP languages in Sect. 6.2, while Sect. 6.3 focuses on CEP systems. The need of processing large amounts of event notifications coming from multiple sources pushed researchers to study how to distribute event processing; we discuss the issues behind this choice in Sect. 6.4. Finally, Sect. 6.5 discusses the most advanced topics in CEP, like management of uncertainty and automated learning of relevant event patterns, and Sect. 6.6 provides some final remarks.

6.2 CEP Languages

This section discusses the main data and processing models for CEP and concretely exemplifies their usage by referring to the TESLA language [7].

6.2.1 Event Model

Event notifications encode interesting occurrences in the domain of analysis [13, 20]. They are characterized by a *time annotation* and a *payload*. The former is used to define ordering relationships among events, while the latter contains relevant information about their occurrence. For example, if an event represents the reading of a vibration from an accelerometer on a painting, then the payload could include the actual acceleration measured, the location of the sensor, and its battery status.

While data stream management systems (DSMSs) assume *homogeneous* streams [10], in which all information items have the same structure, most CEP systems process *heterogeneous* streams of input events. Accordingly, most CEP systems assume event notifications to be annotated with a *type*, which implicitly defines a structure for the payload. Referring to the previous example, a system could define a type `Vibration` for all the vibration occurrences, prescribing a payload of four fields, the first one of type `double` (the acceleration value), the second and third ones of type `string` (the identifier of the room and painting), and the last one of type `int` (the battery percentage). Several *formats* have been proposed to encode the payload of events, ranging from tuples, to key value pairs, to structured XML documents, and to RDF triples [15].

Even more significantly, several *time models* have been discussed in the literature. First of all, the time model specifies the *semantics* of time annotations. There are two common semantics [10, 30]: time annotations may be assigned based on the time indicating when an event enters the CEP system or based on some application time when events have been generated. In the former case, it is easy to define a total ordering among events. The latter case presents more difficulties, since the information items can be received out of order due to unsynchronized application clocks at sources, network latency, and non-order-preserving communication channels.

Furthermore, the time model specifies the *encoding* of time. Most systems adopt a single timestamp, which represents a unique point in time in which the event occurs. Other systems assume that events can have a duration and adopt an interval-based representation, where two timestamps are used, indicating the lower and upper bounds of the interval of occurrence. The use of time intervals enables the definition of a rich set of temporal relations. For example, an interval I_1 can follow an interval I_2, start or finish together with I_2, and overlap or include I_2. An extensive study of the relations between time intervals is present in the pioneering work of Allen [4]. As discussed in [33], different semantics can be provided to define the temporal relations between time intervals (e.g., to define the immediate successor of an item), each of them satisfying different properties (e.g., associativity).

As a concrete example, in the following we will refer to TESLA [7], the language of the T-Rex CEP system [8]. TESLA encodes the payload of events using attribute-value pairs. Each TESLA event is characterized by a type which defines the number, order, names, and types of the attributes that build the notification. Moreover, TESLA assumes that events occur instantaneously at some points in time and encode their time of occurrence into a single timestamp. Referring to the example above, TESLA would encode a vibration reading at time `10` from painting `P` located at room `R` as follows:

```
Vibration@10(value=4.6, room='R', painting='P', battery=85)
```

6.2.2 Processing Model

Although CEP is a relatively new area of research, several CEP systems have been developed in the last few years, each one proposing a different processing model. Nevertheless, it is possible to devise some key commonalities among such models. Indeed, at an abstract level, all systems are based on rules that define composite events starting from patterns of primitive events observed from the environment under analysis. Rules do not explicitly provide the processing steps to be performed; on the contrary, the computation is implicitly specified by the pattern.

The set of operators that allowed inside patterns changes from system to system, but some of them are common: *selection* of primitive events relevant for processing based on their payload; *combination* of multiple events based on their mutual relations in terms of payload and time; *negation*, to identify events that must *not* occur in order to satisfy the pattern; *aggregation* of payload data from multiple primitive events; and *production* of new (composite) events. In the remainder of this section, we present these operators in detail using TESLA as the reference language.

Selection and Combination Rule R1 below *selects* and *combines* two primitive events (`Vibration` and `PeopleNear`) to define a composite event `Touch` that expresses the potential touch of a given painting. In TESLA the occurrence of a composite event is always bound to the occurrence of an observed event (Vibration in our example), which implicitly determines the time at which the new event is detected. This anchor point, called the *terminator* of the rule, is coupled with other events (PeopleNear in our example) through *combination* operators (each-within in our example).

Listing 6.1 Rule R1

```
define  Touch(room: string, painting: string)
from    Vibration(painting=\$p and value>3.0) and
        each PeopleNear(painting=\$p) within 2 min. from ↩
            Vibration
where   room=Vibration.room and painting=Vibration.painting
```

The *selection* operator restricts valid Vibration events based on their value attribute (which must be greater than 3.0). The combination operator binds together Vibration and PeopleNear notifications that (1) refer to the same painting (through the *parameter* constraint painting=$p) and (2) occur within 2 min from each other, with PeopleNear preceding Vibration.

Rule R1 adopts the each-within combination operator, which potentially generates multiple, simultaneous composite events for each terminator (e.g., a Touch event for each PeopleNear event that matches the occurring Vibration event). TESLA also provides other composition operators, e.g., the last-within and first-within operators would generate a single Touch event for every Vibration event, binding it to the last or the first PeopleNear event that satisfies the prescribed time and content constraints. The where clause is used to define the values of the attributes for the produced Touch events.

It is worth noticing that different systems provide different trade-offs between expressiveness and processing efficiency. In particular, some systems define simpler semantics for combination, e.g., by restricting combination to contiguous events [5].

Negation Rule R2 below exemplifies the use of a *negation*, demanding a StaffAt event not to occur in order to distinguish between touches and potential theft.

Listing 6.2 Rule R2

```
define  Theft(room: string, painting: string)
from    Vibration(room=\$r and painting=\$p and value>3.0) and
        each PeopleNear(painting=\$p) within 5 min. from ↩
            Vibration
        and not StaffAt(room=\$r) within 1 min. from Vibration
where   room=Vibration.room and painting=Vibration.painting
```

Aggregation Rule R3 below exemplifies the use of an *aggregation*, which computes the average value over all the `Vibration` events received in the last 5 min to decide if a potential theft is occurring.

Listing 6.3 Rule R3

```
define   Theft(room: string, painting: string)
from     Vibration(room=\$r and painting=\$p and value>3.0) and
         v=Avg(Vibration(room=\$r and painting=\$p)
               within 5 min. from Vibration).value > 3.0
where    room=Vibration.room and painting=Vibration.painting
```

Hierarchies of Events As a final remark, we notice that several systems, including TESLA, enable composite events to take part of patterns that define other composite events. This allows users to build hierarchies of events, where (intermediate) composite events can be used to define other (higher-level) composite events. By allowing output events to reenter the system, this feature enables the definition of recursive rules. Other systems provide a similar expressivity through ad hoc operators that specify trends or unbounded repetitions of events (e.g., all vibration events having values that increase over time) [2].

6.3 Processing Algorithms

This section provides an overview of the main data structures and algorithms used in CEP systems. Due to lack of space, we mainly focus on combination operators, which constitute the core abstraction offered by CEP systems. The interested reader can find additional details in [8, 9, 21]. We present and compare two different approaches, one based on automata and incremental processing (Sect. 6.3.1) and one based on columns and delayed processing (Sect. 6.3.2). Furthermore, we describe how the latter approach can be accelerated by exploiting the parallelism offered by modern hardware architecture (Sect. 6.3.3).

6.3.1 Automata-Based Processing

The main goal of a processing algorithm is to detect patterns of events organized into temporal sequences and also satisfy additional constraints involving their payload (e.g., selection and parameter constraints). At a high level, this problem shares many similarities with the problem of detecting regular expressions into streams of characters, which is usually solved using automata. Because of this, several existing systems, both from academia [1, 5, 29] and from industry,[1] adopt automata-based approaches to process events. In the following, we show the main concepts behind such approaches by describing the AIP (Automata Incremental Processing) algorithm used by T-Rex [8] to deal with TESLA rules.

When adopting the AIP algorithm, each TESLA rule is translated into an *automaton model*, which is composed of one or more *sequence models*. At the beginning, each sequence model is instantiated in a *sequence instance* (or simply *sequence*). Upon a new event arrival, three actions may be taken: (1) new sequences can be created by duplicating existing ones; (2) existing sequences can be moved from a state to the following one; (3) existing sequences can be deleted, either because they arrive to an accepting state or because they are unable to proceed any further.

Creation of Automata As described in Sect. 6.2, each TESLA rule filters event notifications according to their type and content and uses the `*-within` operators to define one or more sequences of events. Let us consider again Rule R1 to exemplify how automata are created. It defines a sequence in which a `PeopleNear` event precedes a `Vibration` event. Furthermore, it requires the value of `Vibration` to be greater than `3.0`, and it forces the two events to refer to the same painting.

Figure 6.1 shows the translation of Rule R1 into an automaton model. AIP creates one *sequence model* for each sequence in the rule (only one in the case of Rule R1). A sequence model is a linear, deterministic, finite state automaton. Each event in the sequence captured by R1 is mapped to a state in the sequence model, and a transition between two states s_1 and s_2 is labeled with the *type*, *content*, and *timing* constraints that an incoming event has to satisfy to trigger the transition. Additional constraints (e.g., parameters) are expressed using dashed lines connecting two states.

Detection Algorithm At the beginning, a single sequence is instantiated from each sequence model built from existing rules. When a new event *e* arrives, the algorithm reacts as follows: (1) it checks whether the type, content, and arrival time of *e* satisfy a transition for one of the existing sequences; if not, the event is immediately discarded. If a sequence `Seq` in a state s_1 can use *e* to move to its next state s_2, the algorithm (2) creates a copy `S'` of `Seq`, and (3) uses *e* to move `S'` to state s_2. Notice that the original sequence `Seq` remains in state s_1, waiting for further events. Sequences are deleted when it becomes impossible for them to proceed to the next

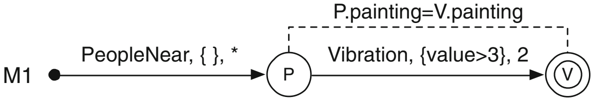


Fig. 6.1 Event detection automata for Rule R1

[1] Esper, http://esper.codehaus.org

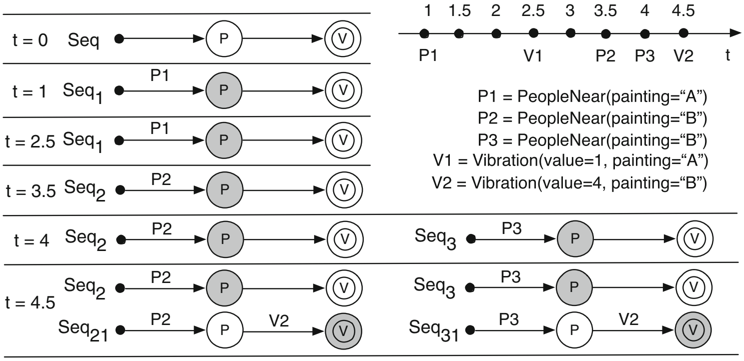

t = 0
Seq
P
V
t = 1
Seq1
P1
t = 2.5
t = 3.5
Seq2
P2
t = 4
Seq3
P3
t = 4.5
Seq21
Seq31
V2
1 1.5 2 2.5 3 3.5 4 4.5
P1 V1 P2 P3 V2 t
P1 = PeopleNear(painting="A")
P2 = PeopleNear(painting="B")
P3 = PeopleNear(painting="B")
V1 = Vibration(value=1, painting="A")
V2 = Vibration(value=4, painting="B")

Fig. 6.2 An example of automata processing

state since the time limits for future transitions have already expired. A sequence in its initial state is never deleted as it cannot expire.

As an example of how processing of sequences works, consider Rule R1 and the corresponding model M1. Figure 6.2 shows, step by step, how the set of incoming events drawn at the upper right corner is processed. At time t=0, a single sequence Seq of model M1 is present in the system, waiting in its initial state. Since Seq does not change with the arrival of new events, we omit it in the figure for t>0. At time t=1, an event P1 of type PeopleNear is received. Since it matches type, content, and timing constraints for the next transition of Seq, we clone it by creating Seq1, which advances to state P. At t=2.5, a Vibration event arrives, but its value is too low to satisfy Rule R1, so it is immediately discarded. At time t=3.5, a new event P2 enters the system. It generates a new sequence Seq2 (cloning Seq) and advances it to state P. At the same time, Seq1 is canceled, since the time limit for its next state transition has expired. At time t=4, event P3 generates a new sequence Seq3 and advances it to state P. At time t=4.5, an event V2 is received. As we are considering a each-within operator, we use V2 to advance every matching sequence in state P. Since V2 satisfies the constraints of both Seq2 and Seq3, it is used to duplicate both of them, generating and advancing two new sequences, Seq21 and Seq31. Seq21 and Seq31 arrive at the accepting state: this means that two valid sequences, composed of events P2–V2 and events P3–V2, have been recognized. This triggers the generation of two composite events. After that, sequences Seq21 and Seq31 are deleted, while Seq2 and Seq3 remain in the system, waiting for potential future events of type Vibration.

As a final remark, we observe that our example adopts the each-within operator. The presence of other composition operators (e.g., last-within) may enable early deletion of sequences, which optimizes the use of resources.

6.3.2 Columns-Based Processing

While the AIP algorithm processes rules incrementally, as new events enter the system, an alternative approach consists in collecting events until a terminator is found. As in the previous case, we present this approach by referring to the CDP (Column Delayed Processing) algorithm implemented into T-Rex, which organizes events into columns, one for each event type that appears in the sequence defined by R, until a terminator is found.

Creation of Columns As an example of how columns are created by the CDP algorithm, consider again Rule R1. For this rule, CDP creates two columns (see Fig. 6.3), each labeled with the type of the primitive events it stores and with the set of constraints on their content. The maximum time interval allowed between the events of a column and those of the previous one (i.e., the window expressed through the `*-within` operator) is modeled using a double arrow. Similarly, additional constraints coming from parameters are represented as dashed lines. Notice that the last column reserves space for a single event.

Detection Algorithm When a new event *e* enters the system, it is processed as follows. First, CDP checks whether it matches (i.e., satisfies type and payload constraints of) one or more columns. If this is the case, *e* is added on top of the matching columns; otherwise it is immediately discarded. If among the matched columns there is the terminating one (c_ℓ), the processing of the events stored so far starts. Processing is performed column by column, from the last one to the first one, creating *partial sequences* of increasing size at each step. More precisely:

- The timestamp of *e* is used to find the index *i* of the first valid element in column $c_{\ell-1}$, by looking at the time window.
- All events in column $c_{\ell-1}$ having an index $i' < i$ are deleted, since they have no chance to enter the window in the future.
- The operation is repeated for each column, considering the timestamp of the first event left in column c_k to delete old events from column c_{k-1}.
- *e* is combined with all the events stored in column $c_{\ell-1}$ that satisfy timing constraints, parameters, and selection policies, creating partial sequences of two events.

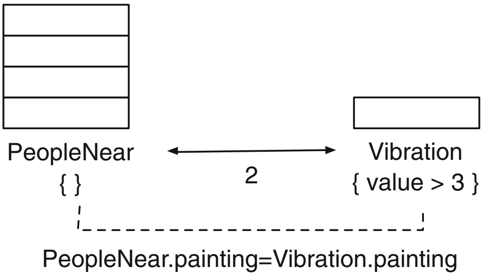


Fig. 6.3 Columns for Rule R1

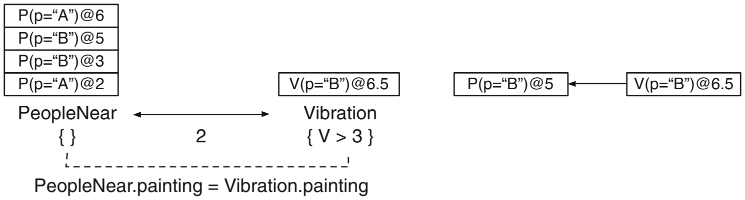


Fig. 6.4 An example of processing using columns

- Each partial sequence is used to select elements from the previous column $c_{\ell-2}$. The algorithm is repeated recursively until the first column is reached, generating zero, one, or more sequences including one selected event from each column.
- One composite event is generated for each sequence.

To better understand how the algorithm works, consider again Rule R1 and the situation in Fig. 6.4, where the events stored in each column are represented with their type, their value for the attribute `painting` (`p`), and their timestamp. The event `V(p='B')@6.5` was the last to enter the system. Since it is a terminator for Rule R1, it starts the processing algorithm. Its timestamp is used to permanently discard elements of the previous column that are too old to satisfy timing constraints. In particular, we discard all events having a timestamp lower than `4.5` (`6.5-2`). This operation is performed recursively in the presence of more than two columns. The remaining events (i.e., `P(p='B')@5` and `P(p='A')@6`) are evaluated to detect valid sequences. The former matches the constraint on the `painting` attribute and is used to create the valid sequence shown on the right of Fig. 6.4. On the contrary, the latter does not match the attribute constraint. Both events are kept in the leftmost column, waiting for the arrival of further `Vibration` events.

6.3.3 Exploiting Parallel Hardware

One of the key benefits of the CDP algorithm consists in the simple layout of its data structures, which makes it suitable for running on parallel hardware architectures. To exemplify this feature, we present an implementation of CDP on Compute Unified Device Architecture (CUDA) graphics processing units (GPUs).

CUDA CUDA is a parallel computing architecture introduced by Nvidia in 2006 to offer a new parallel programming model and instruction set for general-purpose programming on GPUs. CUDA has been exploited in several domains to speed up complex computations. However, attaining good performance with CUDA is challenging, and only some algorithms can be effectively ported to such parallel architecture. Next, we briefly discuss the main features of CUDA and how they affect the design of algorithms. While we focus on CUDA, most of the concepts presented below apply in general to several modern parallel architectures (e.g., Xeon Phi) and programming API (e.g., OpenMP).

The CUDA programming model assumes that CUDA threads execute on a *device* (the GPU), which operates as a coprocessor to a *host* (the CPU) and has its own separate memory space. The programmer starts a new computation on the device by invoking a function, called *kernel*, which defines a single flow of execution. CUDA implements a Single Program Multiple Threads paradigm: the kernel is executed in parallel by a number of threads on different data. Threads can be combined in (multidimensional) groups. Inside the kernel, each thread is identified by a *groupId* and a *threadId* variable. Conditional statements involving these variables are the only way for a programmer to differentiate the execution flows of different threads.

There are two main factors to consider while programming in CUDA, which can severely impact on performance. (1) A modern GPU can run thousands of threads concurrently, but its hardware is designed for data parallel executions. Because of this, full efficiency is achieved only if all the threads agree on their execution path. If threads diverge via a data-dependent conditional branch, CUDA serially executes each branch path taken. (2) Often, retrieving data from memory constitutes the main bottleneck for CUDA. To alleviate this issue, it is necessary to design the data structures in such a way that threads with contiguous threadId access contiguous memory regions. This enables the hardware to fully exploit the available memory bandwidth.

Accelerating CDP with CUDA The complexity of AIP's data structures prevents the possibility of efficiently implementing it in CUDA. Accordingly, only CDP has been ported to CUDA.

In CUDA, the GPU memory is pre-allocated by the CPU and this operation has a non-negligible latency. Because of this, CDP implements each column as a statically allocated circular buffer. Furthermore, it keeps a copy of each column into the CPU memory, which allows to perform on the CPU all the operations that do not benefit from parallelization. In particular, when an event *e* enters the system and matches a state *s* for a rule *r*, it is added to the column for *s* in main memory. Then, a copy of the event to the GPU memory is issued asynchronously, such that the CPU does not need to wait for the copy to end. If *e* is a terminator for *r*, the CPU computes which events need to be considered for each column, starting from their timestamp and from the time windows in *r*. CDP performs this operation on the CPU since it requires a sequential analysis of the columns (from the last to the first one), and the computation inside each column can be efficiently performed by a standard CPU through a binary search. At the end of this phase, the CPU invokes the GPU to process the relevant events from each column.

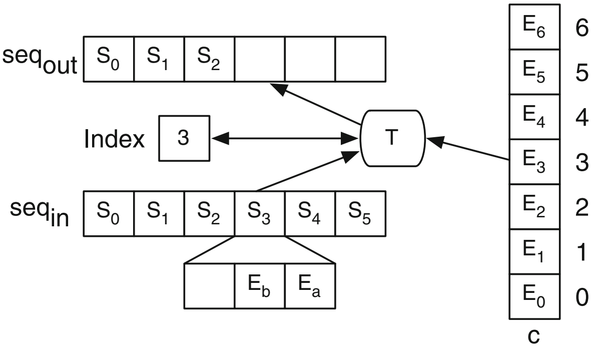
When using the `each-within` operator to process a column *c*, each partial sequence generated at the previous step may be combined with more than one event in *c*. The algorithm performs the following steps:

- It allocates two arrays of sequences, called seq_{in} and seq_{out}, used to store the input and output results of each processing step. Sequences are represented as fixed-size arrays of events, one for each state defined in the rule.

- It allocates an integer *index* and sets it to 0.
- At the first step, seq_{in} contains a single sequence with only the last position occupied (by the received terminator).
- When processing a column c, a different thread t is executed for each event e in c and for each sequence *seq* in seq_{in}.
- t checks if e can be combined with *seq*, i.e., if it matches timing and parameter constraints of *seq*.
- If all constraints are satisfied, t uses a special CUDA operation to atomically read and increase the value of *index*. The read value k identifies the first free position in the seq_{out} array: the thread adds e to *seq* in position c and stores the result in position k of seq_{out}.
- When all threads have finished, the CPU copies the value of *index* into the main memory and reads it.
- If the value of *index* is greater than 0, it proceeds to the next column by resetting the value of *index* to 0 and swapping the pointers of seq_{in} and seq_{out} into the GPU memory.
- The algorithm continues until *index* becomes 0 or all the columns have been processed. In the first case, no valid sequence has been detected, while in the second case, all valid sequences are stored in seq_{out} and can be copied back to the CPU memory.

To better understand how the CDP algorithm works, consider the example in Fig. 6.5. It shows the processing of a rule R defining a sequence of three primitive events. Two columns have already been processed resulting in six partial sequences of two events each, while the last column c to be processed is shown in the figure. Since there are six sequences stored in seq_{in} and seven events in c, the computation requires 42 threads. Figure 6.5 shows one of them, thread T, which is in charge of processing the event E_3 and the partial sequence S_3. Now suppose that E_3 satisfies all the constraints of Rule R and thus can be combined with S_3. T copies E_3 into the first position of S_3; then, it reads the value of *index* (i.e., 3) and increases it. Since this operation is atomic, T is the only thread that can read 3 from *index*, thus

Fig. 6.5 CDP algorithm on CUDA (each-within operator)



avoiding memory clashes when it writes a copy of S_3 into the position of index 3 in seq_{out}.

In this implementation, threads with contiguous identifiers are used to process contiguous positions in a column: this increases the performance of memory access, since the hardware can combine operations issued by different threads.

6.3.4 Performance Analysis

This section provides an overview of the performance of the algorithms described above. All the results discussed below have been collected using a 2.8 GHz AMD Phenom II PC, with 6 cores and 8 GB of DDR3 RAM, running 64 bit Linux.

Figure 6.6 compares the performance of the CPU implementation of AIP and CDP. We deploy 1000 rules in the system, all of them defining a sequence composed of a variable number of events. Each incoming event is relevant for exactly 1 % of the deployed rules; each rule (and each state inside a rule) has the same probability to select incoming events. We consider an average window size of 15 s between two consecutive events in a sequence.

First, Fig. 6.6 highlights the efficiency of both the AIP and the CDP algorithms. We consider two scenarios: the first adopts the `each-within` operator to define sequences (Fig. 6.6a), while the second adopts the `last-within` operator (Fig. 6.6b). In both cases, the considered algorithms can easily process events in sub-millisecond time. Second, as expected, managing the `each-within` operator is more expensive—more primitive events need to be combined and more composite events are generated—and results in higher processing times for both algorithms. Third, Fig. 6.6 shows that a traditional approach based on automata is less efficient than the CDP algorithm.

As a second step, we performed an additional experiment to test the benefits of adopting massively parallel hardware. In this experiment, we deploy a single rule r defining a sequence of length 3. We repeat the measurement with different sizes of windows, which is probably the parameter that affects a GPU implementation the most. Larger window sizes increase the average number of events to be considered

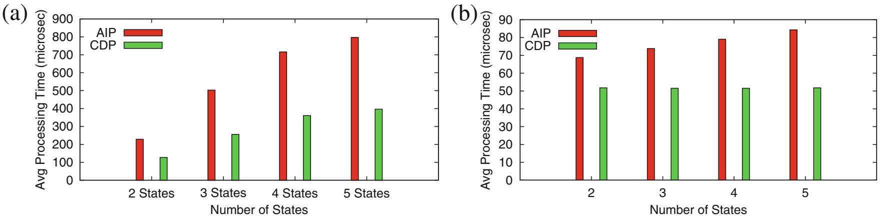


Fig. 6.6 Comparison of automata-based and column-based algorithms. (**a**) Each-within. (**b**) Last-within

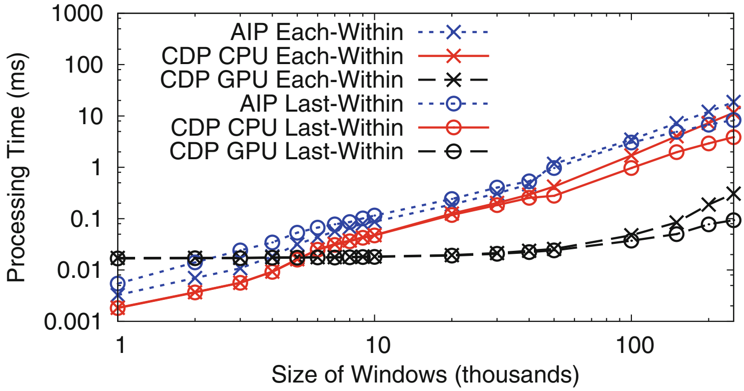


Fig. 6.7 Analysis of the benefits of parallel hardware

at each state. While the CPU processes those events sequentially, the GPU uses different threads running in parallel. Furthermore, using the GPU involves an (almost constant) overhead to transfer input data from the CPU to the GPU memory and to activate a CUDA kernel. This makes the usage of the GPU convenient only if a significant number of events to be processed is present at each state.

Figure 6.7 emphasizes this behavior: on one hand, the cost of the algorithms running on the CPU grows with the size of windows. On the other hand, the cost of the CDP algorithm running on the GPU is initially constant at 0.017 ms (it is dominated by the fixed overhead of CUDA), and it starts growing only when the number of available threads is not enough to compute events entirely in parallel. This growth is faster with the `each-within` operator, which uses more threads and produces more composite events to be transferred back to the main memory. In our test, the smallest size of windows that determines an advantage in using the GPU is 4000. With a sequence of three states, this results in considering 1333 events in each state, on average. This result isolates one of the most significant dimensions for deciding the hardware architecture to adopt. If an application involves rules that need to process a small number of events for each state, then the CPU represents a better choice. Otherwise, the advantages of parallel processing favor the use of a GPU.

6.4 Protocols for Distributed Event Detection

In the previous sections, we assumed that primitive events were delivered to a single node responsible for detecting composite events and delivering them to the interested recipients. However, several application domains may involve components dispersed over a wide geographical area. This is certainly the case for many pervasive systems. In these settings, centralizing the processing in a single node is not ideal, since it prevents from exploiting the locality of information: even if a (small) group of nodes holds all the information required to detect a composite event, it still needs to rely on the centralized processing server and cannot operate autonomously. This increases the time for detecting composite events and the amount of data exchanged through the network, which, in turn, negatively impacts the lifetime of battery-powered devices. Finally, a centralized solution may hamper scalability when the processing resources of the processing node are not sufficient.

6.4.1 Distribution Strategies

To overcome the limitations described above, some systems proposed *distribution strategies* to enable a decentralized processing of rules. A distribution strategy involves (1) an algorithm to decide how the processing load is partitioned among different processing nodes and (2) a communication protocol that defines how the nodes interact and communicate with each other to produce the required results.

The first algorithm solves the *operator placement* problem: it searches for the best mapping of the operators defined in rules on the set of available nodes. Depending on the application, the operator placement may pursue different goals, e.g., reducing the latency to detect and notify composite events or minimizing the usage of network resources. Given the complexity of this problem, several works addressed it using approximated algorithms and heuristics [19]. They usually rely on a centralized decider, which collects information about the status of the nodes and locally computes a suitable deployment of operators. Only a few solutions considered decentralized algorithms [25]. Finally, most operator placement algorithms are studied for cluster infrastructures, in which all processing nodes are colocated and well connected [17, 34]: in this setting, the operator placement problem essentially translates into a load balancing problem.

Besides operator placement, a distribution strategy also requires a communication protocol to govern the interaction among processing nodes, specifying how rules are deployed and how primitive and composite events are forwarded. These issues are rarely considered by existing CEP systems: even when distributed processing is allowed, the communication among nodes requires manual configuration [3].

In the remainder of this section, we briefly introduce some concrete examples of distribution strategies for the T-Rex system [11].

6.4.2 A Concrete Example: Distribution Strategies for T-Rex

To illustrate possible distribution strategies for T-Rex, we consider a set of processing nodes P connected with each other in a physical network. To simplify the routing, our deployment strategies organize nodes into one or more *processing trees* on top of the physical network. More precisely, they use a processing tree to collect primitive events from sources (the leaves) and to filter and (partially) process them as they move toward the root of the tree, where the composite events are generated. This enables incremental evaluation of rules at intermediate nodes, reducing the amount of information flowing along the tree and the processing load at the root node. The same tree adopted for detecting a composite event *ce* is also used to distribute *ce* to the interested clients. Since we want to minimize latency, we build *Shortest Path Trees* using the link delay as a cost metric.

Single Tree vs. Multiple Trees We consider two classes of deployment strategies. The first one organizes all nodes into a single processing tree. One node ℓ is elected as the network leader and all events move from the sources to ℓ going up along the tree rooted at ℓ (T_ℓ), while they get incrementally evaluated according to the rules deployed in the system. When they reach ℓ, the processing is complete, and the corresponding composite events are generated and delivered to the sinks along T_ℓ.

The second class of strategies builds one tree for each client interested in a composite event (for each *sink*). Primitive events flow from the sources along multiple trees. In particular, if a sink s is interested in a composite event *ce*, all primitive events that contribute to *ce* move from their sources up along T_s. Processing is performed incrementally on each tree. When a composite event reaches the root of a processing tree, there is no need to further distribute it, since the root of the tree coincides with the interested client. This approach removes the need for spreading composite events once they have been detected at the cost of duplicating primitive events that must be forwarded over multiple trees.

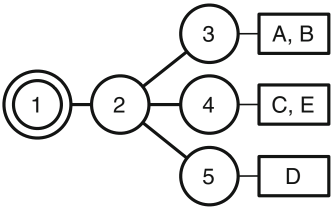
Partitioning TESLA Rules To enable incremental evaluation on a processing tree T, rules are recursively partitioned into *partial rules* moving from the root to the leaves of T. The goal is to push the processing of events as close as possible to the sources where events are generated. To do so, nodes first declare the type of events they will produce (e.g., an accelerometer will produce only `Vibration` events). This knowledge is used for partitioning: the parts of a rule responsible for detecting a certain event *e* are deployed as close as possible to the nodes that produce *e*.

Next, we offer an overview of the partitioning algorithm. The interested reader can refer to [6, 11] for further details. Let us consider Rule R4 and the processing tree T_1 (Fig. 6.8), rooted at 1 and containing three sources: 3 produces events of types `A` and `B`; 4 produces events of types `C` and `E`; 5 produces events of type `D`. This information is available to node 2. Similarly, node 1 knows that it can receive events of types `A`, `B`, `C`, `D`, and `E` from node 2.

Listing 6.4 Rule R4

```
define     CompEvent()
from       A() and last B() within 5 min. from A
           and last C() within 5 min. from B
           and last D() within 5 min. from C
           and last E() within 5 min. from D
```

Fig. 6.8 Rule deployment: an example



Partitioning is performed as follows: 1 observes that 2 receives all the information necessary to correctly evaluate the rule. Accordingly, it entirely delegates the processing of $R4$ (including the generation of composite events) to 2. 2 observes that none of its children has enough information to process Rule R4. Accordingly, 2 remains responsible for producing composite events while it delegates only parts of the processing to 3, 4, and 5 in the form of partial rules.

As mentioned, partial rules are used to filter primitive events as close as possible to sources. A node p responsible for a partial rule r' forwards a set of primitive events to its parent in the processing tree only when this satisfies the pattern in r'. Let us consider node 3: its clients are the only sources of events `A` and `B`. To correctly process Rule R4, node 2 does not need to receive all events of types `A` and `B` but only events `A` that are preceded by an event `B` in the previous 5 min; moreover, only the last `B` event before each `A` is relevant (Rule R4 uses the `last-within` operator). Accordingly, 2 creates the following partial rule for 3:

```
A() and last B() within 5 min. from A
```

Similarly, 2 does not need to receive all `C` events but only those preceded by an event of type `E`. Accordingly, it creates and sends the following partial rule to 4:

```
C() and each E() within 10 min. from C
```

Notice that `C` and `E` are not contiguous elements in the sequence defined by Rule R4, but they are separated by event `D`. Because of this, the partial rule considers a window that sums the one between `C` and `D` and the one between `D` and `E`. Similarly, the local knowledge of node 4 is not sufficient to evaluate the single selection constraint on `E`; for this reason, the partial rule adopts the `each-within` operator, capturing all notifications of `E` followed by a `C` event within 10 min. Finally, node 5 receives a partial rule that simply asks for all events of type `D`.

The partitioning algorithm described above is applied recursively: partial rules are split into other partial rules until all sources have been reached.

Forwarding of Events Once detected, a primitive event p is forwarded along the relevant trees, i.e., those associated to rules that involve p (a decision taken by looking at the type of p). In some cases, this *push-based forwarding* strategy can be complemented by a *pull-based forwarding* strategy that tries to limit network traffic by pulling events of a certain type only when other events they are related to have been detected. As an example, considering Rule R1 described in Sect. 6.2, it may be better to avoid forwarding PeopleNear events until a (less frequent) Vibration event is detected (for further details, see [11]).

6.4.3 Performance Analysis

In the following, we report some measurements conducted in a simulated environment to show the benefits of distributed processing on reducing the network traffic and the delay for delivering events. In particular, we consider three different strategies: ST performs distributed processing on a single tree; MT performs distributed processing on multiple trees; Centr exploits a single processor, which receives primitive events, processes them, and delivers composite events to interested sinks.

Tests are performed in a scenario that includes 20 nodes, each one connected with five others, on average. Sources produce 120 different types of primitive events with a generation rate between 1000 s and ten notifications per second, with exponential distribution. We deploy 100 TESLA rules, each one including a sequence of three events with time windows of 1 min, on the average. Each rule produces a different composite event, and a set of sinks connected with each processor is interested in ten of them, on average.

Figure 6.9a shows the delay for delivering events. First, we observe that MT strategies provide much lower delays with respect to ST strategies. Indeed, MT strategies do not need to deliver composite events after detection, thus eliminating the delay introduced in this phase. ST strategies perform similarly to a centralized scenario: both need to deliver composite event notifications after detection.

Network traffic is significantly higher in the Centr strategy (Fig. 6.9b). This means that distributed processing effectively enables the filtering of a large number of primitive events close to their sources. If we compare the two distributed strategies, we observe that MT generates more traffic than ST. While the former does not require the forwarding of composite events after detection, it demands for the forwarding of primitive events along multiple trees.

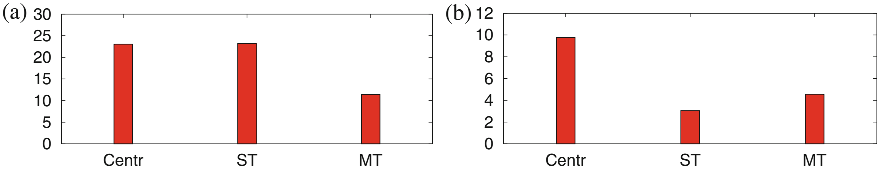


Fig. 6.9 A comparison of distribution strategies for T-Rex. (**a**) Average delay (ms). (**b**) Average traffic (KB/s)

6.5 Advanced Topics

This section introduces some advanced topics in the CEP domain that are currently investigated in research. It focuses on three topics relevant for pervasive systems: management of uncertainty, automated generation of CEP rules, and integration with programming languages.

Management of Uncertainty CEP technologies are used to model and capture phenomena of interest. In this context, the accuracy in modeling the domain of analysis is fundamental for a successful adoption of CEP. At the same time, human ability to define and capture a phenomenon is often affected by some form of uncertainty, which, if ignored, may lead to incomplete, inaccurate, or even incorrect decisions concerning the phenomenon itself.

A successful management of uncertainty in CEP involves three main steps: *identification* of the sources of uncertainty, *modeling* of uncertainty, and *propagation* of uncertainty across the systems, from primitive to composite events.

The identification step requires an analysis of the environment and of the specific phenomena to capture. In general, we can identify two main sources of uncertainty [12]: uncertainty in the input data and uncertainty in the CEP rules. The former refers to the presence of incomplete, incorrect, or imprecise information observed from the external environment. This is extremely relevant for pervasive systems. For example, in the case of input data coming from sensors, the value measured and propagated may be imprecise (because of the limited accuracy in the sensor) or incomplete (because of communication or battery-related issues). The latter refers to the imprecision of the rules in correctly formulating the causal relations between the primitive and the composite events. This may derive from a limited knowledge of the environment under analysis but also from the impossibility to observe all the aspects that may influence the occurrence of a phenomenon.

The modeling step aims at providing a sound mathematical foundation to represent uncertainty, let the CEP engine be aware of it, and manipulate it consistently. For example, probability theory can be used to model measurement errors, allowing the CEP engine to process uncertain values and combine them with other ones.

Finally, an uncertainty-aware CEP system should propagate the uncertainty by producing results that are annotated with uncertainty information consistent with the identified sources of uncertainty and the models adopted to represent them.

The recent literature presents several proposals for capturing uncertainty in CEP [12, 18, 27, 31, 32]. Most of these solutions rely on an explicit encoding and representation of uncertainty inside data items. For instance, [12] adopts random variables to represent received information, thus making it possible to encode measurement errors in sensor applications.

There are various metrics for evaluating a model for uncertainty, including expressiveness, precision, computational cost, and simplicity. While end consumers may be interested in receiving precise indications about the level of certainty associated to the results, this should not negatively impact the compactness and readability of information or the level of performance of the CEP system.

Automated Generation of Rules Our discussion of uncertainty highlights key issues of current CEP systems: while research and development efforts were mainly directed toward processing efficiency, a widespread adoption of CEP technologies depends on the capability to correctly and precisely model the phenomena under analysis. As our discussion in Sect. 6.2 shows, this task can be extremely difficult and involve several different aspects, including identification of the relevant primitive events, filtering of their content, and identification of their mutual relation in terms of content and time ordering.

To overcome this problem, researchers are currently focusing on defining techniques to support users in rule definition. In particular, some preliminary results have been achieved in the area of learning CEP rules from the available historical information about the environment under analysis [23]. These techniques adopt automated algorithms that analyze past occurrences of the phenomena of interest and suggest CEP rules for detecting them.

Language Integration As discussed so far, CEP represents a mainstream technology for promoting the interaction of components in complex software systems, typical in many scenarios of pervasive systems. In this context, CEP becomes a key component for *programming* the overall behavior of the software architecture. Because of this, some research efforts targeted the integration of event processing within programming languages and frameworks.

In this context, we can distinguish two main areas of investigation. On the one hand, some languages have been defined that provide primitives to send and receive events. In these proposals, events become first-class objects of the language and are fully integrated, e.g., with the type system. Furthermore, some CEP operators for event composition and pattern detection are offered as language construct. Examples of this approach are EventJava [14], Ptolemy [26], and EScala [16].

On the other hand, some systems are built on top of event processing to define a novel programming paradigm known as *reactive programming*. In reactive programming, developers can define *reactive* variables through an expression that involves other (reactive or imperative) variables. Whenever one variable changes, all dependent variables get automatically updated. While notifications of changes are implemented as events, they are not exposed to programmers, who only observe their effect (i.e., changes to the values of reactive variables). Examples of reactive systems are REScala [28], Flapjax [24], and DREAM [22].

6.6 Conclusions

This chapter provided an overview of CEP technology, considering both the operators it offers to applications and some key algorithms and implementation mechanisms used to achieve high-performance processing and scalability.

In the domain of pervasive systems, CEP represents an ideal solution to guide the interactions of a plethora of distributed components. Indeed, CEP enables domain experts to focus on the modeling of the application domain and to entirely delegate to the CEP system the task of observing and responding to the stimuli of the application environment according to such modeling.

References

1. Adi, A., Etzion, O.: Amit - the situation manager. VLDB J. **13**(2), 177–203 (2004)
2. Agrawal, J., Diao, Y., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD ‘08, pp. 147–160. ACM, New York (2008)
3. Ali, M.: An introduction to microsoft sql server streaminsight. In: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application, COM.Geo ‘10, pp. 66:1–66:1. ACM, New York (2010)
4. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
5. Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M., White, W.: Cayuga: a high-performance event processing engine. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD ‘07, pp. 1100–1102. ACM, New York (2007)
6. Cugola, G., Margara, A.: Raced: an adaptive middleware for complex event detection. In: Proceedings of the 8th International Workshop on Adaptive and Reflective MIddleware, ARM ‘09, pp. 5:1–5:6. ACM, New York (2009)
7. Cugola, G., Margara, A.: Tesla: a formally defined event specification language. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS ‘10, pp. 50–61. ACM, New York (2010)
8. Cugola, G., Margara, A.: Complex event processing with t-rex. J. Syst. Softw. **85**(8), 1709–1728 (2012)
9. Cugola, G., Margara, A.: Low latency complex event processing on parallel hardware. J. Parallel Distrib. Comput. **72**(2), 205–218 (2012)
10. Cugola, G., Margara, A.: Processing flows of information: from data stream to complex event processing. ACM Comput. Surv. **44**(3), 15:1–15:62 (2012)
11. Cugola, G., Margara, A.: Deployment strategies for distributed complex event processing. Computing **95**(2), 129–156 (2013)
12. Cugola, G., Margara, A., Matteucci, M., Tamburrelli, G.: Introducing uncertainty in complex event processing: model, implementation, and validation. Computing **97**(2), 103–144 (2015)
13. Etzion, O., Niblett, P.: Event Processing in Action, 1st edn. Manning Publications Co., Greenwich (2010)
14. Eugster, P., Jayaram, K.: Eventjava: an extension of java for event correlation. In: Drossopoulou, S. (ed.) ECOOP 2009 – Object-Oriented Programming. Lecture Notes in Computer Science, vol. 5653, pp. 570–594. Springer, Berlin/Heidelberg (2009)
15. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Comput. Surv. **35**, 114–131 (2003)
16. Gasiunas, V., Satabin, L., Mezini, M., Núñez, A., Noyé, J.: Escala: modular event-driven object interactions in scala. In: Proceedings of the Tenth International Conference on Aspect-Oriented Software Development, AOSD ‘11, pp. 227–240. ACM, New York (2011)
17. Khandekar, R., Hildrum, K., Parekh, S., Rajan, D., Wolf, J., Wu, K.L., Andrade, H., Gedik, B.: Cola: optimizing stream processing applications via graph partitioning. In: Middleware ‘09, pp. 1–20. Springer, New York (2009)

18. Kuka, C., Nicklas, D.: Quality matters: supporting quality-aware pervasive applications by probabilistic data stream management. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, pp. 1–12. ACM, New York (2014)
19. Lakshmanan, G.T., Li, Y., Strom, R.: Placement strategies for internet-scale data stream systems. IEEE Internet Comput. **12**(6), 50–60 (2008)
20. Luckham, D.: The power of events: an introduction to complex event processing in distributed enterprise systems. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) Rule Representation, Interchange and Reasoning on the Web. Lecture Notes in Computer Science, vol. 5321, pp. 3–3. Springer, Berlin/Heidelberg (2008)
21. Margara, A.: Combining expressiveness and efficiency in a complex event processing middleware. Ph.D. thesis, Politecnico di Milano (2012)
22. Margara, A., Salvaneschi, G.: We have a dream: distributed reactive programming with consistency guarantees. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, pp. 142–153. ACM, New York (2014)
23. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the past: automated rule generation for complex event processing. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, pp. 47–58. ACM, New York (2014)
24. Meyerovich, L.A., Guha, A., Baskin, J., Cooper, G.H., Greenberg, M., Bromfield, A., Krishnamurthi, S.: Flapjax: a programming language for ajax applications. In: Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA '09, pp. 1–20. ACM, New York (2009)
25. Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., Seltzer, M.: Network-aware operator placement for stream-processing systems. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, p. 49. IEEE Computer Society, Washington (2006)
26. Rajan, H., Leavens, G.: Ptolemy: a language with quantified, typed events. In: Vitek, J. (ed.) ECOOP 2008 – Object-Oriented Programming. Lecture Notes in Computer Science, vol. 5142, pp. 155–179. Springer, Berlin/Heidelberg (2008)
27. Ré, C., Letchner, J., Balazinksa, M., Suciu, D.: Event queries on correlated probabilistic streams. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pp. 715–728. ACM, New York (2008)
28. Salvaneschi, G., Hintz, G., Mezini, M.: Rescala: bridging between object-oriented and functional style in reactive applications. In: Proceedings of the 13th International Conference on Aspect-Oriented Software Development, AOSD, vol. 14 (2014)
29. Schultz-Møller, N.P., Migliavacca, M., Pietzuch, P.: Distributed complex event processing with query rewriting. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS '09, pp. 4:1–4:12. ACM, New York (2009)
30. Srivastava, U., Widom, J.: Flexible time management in data stream systems. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems, pp. 263–274. ACM, New York (2004)
31. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Complex event processing over uncertain data. In: Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS '08, pp. 253–264. ACM, New York (2008)
32. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Efficient processing of uncertain events in rule-based systems. IEEE Trans. Knowl. Data Eng. **24**(1), 45–58 (2012)
33. White, W., Riedewald, M., Gehrke, J., Demers, A.: What is "next" in event processing? In: PODS, pp. 263–272. ACM, New York (2007)
34. Wolf, J., Bansal, N., Hildrum, K., Parekh, S., Rajan, D., Wagle, R., Wu, K.L., Fleischer, L.: Soda: an optimizing scheduler for large-scale stream-based distributed computer systems. In: Middleware '08, pp. 306–325. Springer, New York (2008)

Chapter 7
Applying Semantic Interoperability Principles to Data Stream Management

Daniele Dell'Aglio, Marco Balduini, and Emanuele Della Valle

7.1 Introduction

The cost versus opportunity trade-off in ICT projects often pushes separate organizations and even departments of some organizations to collect and manage data independently. This creates the so-called data silos. However, early success of those ICT projects commonly results in requests for cross-silo usages of data and, therefore, for addressing the data integration problem [28]. Systems addressing this problem have to bridge the silos and produce a uniform query interface.

Data streams do not make an exception. Consider, for instance, the case of a large cultural heritage site where two successful ICT projects deployed near-field communication (NFC) sensors [31] to let the guide signal where they are and a quick response[1] (QR) code-based system that tourists can use to obtain information about a part of the exposition. Add to this scenario the common habit of tourists to share on social networks their impressions about the visiting experience. These three streaming data sources are data silos. They serve the purpose of the application they were built for. However, it is easy to imagine the value of cross-silo usage of these data streams. For instance, it would be possible to recommend to a free guide to propose an attractive short guided tour to the tourists nearby based on the preferences they expressed by using the QR codes and sharing snippets of their visits on social networks.

[1] BS ISO/IEC 18004:2006. Information technology. Automatic identification and data capture techniques. Bar code symbology. QR Code. Geneva: ISO/IEC. 2000. p. 114.

D. Dell'Aglio (✉) • M. Balduini • E.D. Valle
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico of Milano, P.za L. Da Vinci 32, I-20133 Milano, Italy
e-mail: daniele.dellaglio@polimi.it; marco.balduini@polimi.it; emanuele.dellavalle@polimi.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_7

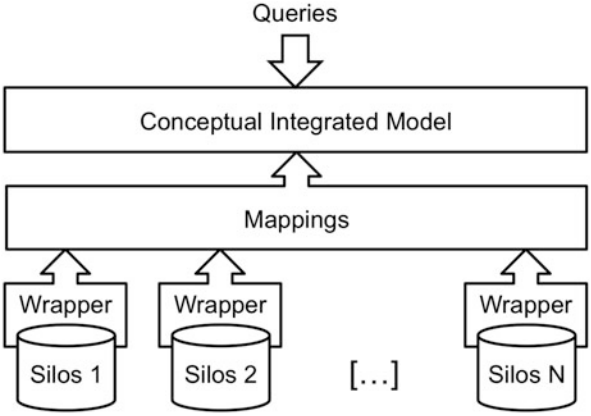


Fig. 7.1 The architecture of data integration systems

The problem of data integration has been studied for decades, but the experimental nature of the data acquisition solutions, which produce data streams, and the poor attention to this type of data, which is often considered too low a level to be useful before aggregation, exacerbate the problem and require a new generation of data integration solution tailored to data streams.

Most data integration systems adopt the architecture outlined in Fig. 7.1. Data in the silos can be organized in relational, extensible markup language (XML) or graph databases and in some cases even in textual and multimedia repositories. In two separate silos, the same information can appear with different syntaxes, data structures, and conceptual models, raising, respectively, the syntactic, structural, and semantic heterogeneity problem. For example, the notion of user is likely to be different for the QR code-based system and a social network. The different business and legal requirements will influence their respective data representations. For example, a social network is unlikely to keep a record of what pieces in the site its users asked information about, while the QR code-based system is unlikely to keep track of what impressions its users shared with their friends.

An **Integrated Conceptual Model** (ICM) is required to bridge the syntactic, structural, and semantic heterogeneities of the data in the silos. In our example, an ICM of the user would consider all information about users available in the various silos. The problem of representing and querying incomplete information is extremely relevant in this context. Consider, for instance, a piece in the site. It is explicitly named when a user asks its description using the QR code-based application. On the contrary, if a user checks in the site using a social network, it is only possible to infer that the user has seen some pieces of the site without knowing exactly which one. In order to develop the ICM, an adequate data modelling language is needed. In the literature, a variety of languages were proposed. They differ for their ability to capture relationship between terms used to describe the data in the silos. In the last decade, ontological languages like ontology Web language (OWL) [29] and OWL 2 [30] were often adopted for this purpose. They can adequately master the semantic heterogeneity between the silos.

The ICM defines a vocabulary to issue (or register in the streaming setting) queries across the silos, but in order to obtain results, we have to link the terms describing the data in the silos to those in the ICM. Expressions to model those links are named **mappings**, and they can assume multiple forms (i.e., Local as View, Global as View, and both [28]). Mappings only master structural heterogeneity assuming the silos and the ICM are requested in the same data model. This is often not the case, so a **wrapping** solution is needed to bridge the syntactic heterogeneity between the data in the silos (e.g., relational) and the data model assumed by the mapping language. In some cases, hybrid solutions that wrap data in silos and map it to the ICM exist, e.g., to map a given data model (e.g., relational) into the data model of the ICM. In recent years, we observed a growing adoption of the resource description format (RDF) [25] as a data model of the ICM because it is adequate to represent data when the ICM is modelled in OWL or OWL 2. When those technologies are adopted and the data in the silos is relational, R2RML [18] is the associate mapping language to be used and SPARQL [33] (or its extensions to data streams [10]) is the language to declare queries with. Dialects of R2RML exist to map between data models that resemble the relational one, e.g., when data are stored in comma-separated value (CSV) files or are XML or JavaScript object notation (JSON) results of Web service invocations. The Any23 solution[2] is a good example of flexible open-source framework to wrap any type of data source as RDF.

The semantics of the data integration solution as a whole—i.e., to define which are the correct answers to a query on top of a given set of data in the silos—is determined by the semantics of: 1) the ontological language used to model the ICM, 2) the mappings, 3) the wrappers, 4) the data model to represent data in the ICM, and 5) the query langue used to declare the queries. Given the complexity of the task of defining the semantics for a given data integration solution as a whole, readers should not be surprised that the current literature lacks a uniform and well-accepted theory that describes the various parts of data integration systems. Only a consistent theory for a particular data integration solution may be found.

This chapter considers only two extreme solutions: data driven and query driven. In data-driven solutions, the ICM is populated with data before starting to answer a query declared on it. Solutions of this kind range from materialized views in databases [23] up to consequence-based reasoning [24, 36]. The limitation of data-driven solutions is the need to duplicate in the ICM all the data in the silos even if only a limited fraction is required to answer a query. On the contrary, in query-driven solutions, a query expressed on the ICM is translated into a set of queries expressed in the query language (if any) of the silos [16, 22]. The queries are then evaluated in the silos, and the results are integrated into an answer to the original query. Unfortunately, the query-driven approach is not always possible and depends on the ontological language used to model the ICM. In recent years, both approaches were explored to integrate data streams (see, for instance, [9] for a data-driven approach and [14] for a query-driven one). They are commonly referred to as embodiments

[2] http://any23.apache.org/

of the Stream Reasoning vision [19], and consensus is growing[3] around the idea to name them RDF Stream Processing (RSP) systems. It is worth noting that porting solutions known to work in the static setting to the streaming one are far from being trivial. Data-driven solutions are at risk of being impractical due to the very unbound nature of data streams, while query-driven ones require more investigation to master the operation semantic heterogeneity that exists in data stream management systems [12].

The remaining part of the chapter is organized as follows: Section 7.2 formalizes the notion of RDF stream and the semantics of continuous query processing of RDF streams. Section 7.3 briefly surveys existing implementations. Section 7.4 assembles all elements introduced in the previous sections and puts them at work on a case study. Finally, Sect. 7.5 concludes and casts some light on future developments in the field.

7.2 RSP Models

The main feature of RDF Stream Processing is its data model: instead of traditional relational data, it uses the RDF data model. This section first introduces the RDF data model, then presents the different data, and proposes query models in the literature.

7.2.1 The RDF Data Model

RDF is a W3C recommendation for data interchange on the Web [25]. RDF data are structured as directed labeled graphs, where the nodes are **resources**, and the edges represent relations among them. Each node of the graph is an **RDF term**: it can be named a **resource** (identified by an Internationalized Resource Identifier (IRI)), an anonymous resource (also known as **blank node**), or a **literal** (a number, a string, a date, etc.). Let us denote with I, B, and L the set of resources, blank nodes, and literals. The basic building block of RDF is the **RDF statement**, a triple $(s, p, o) \in (I \cup B) \times (I) \times (I \cup B \cup L)$. A set of RDF statements is an **RDF graph**. Let us consider the following chunk of RDF data (Listing 7.1), extracted by DBPedia[4].

[3] http://www.w3.org/community/rsp/

[4] Cf. http://dbpedia.org

Listing 7.1 RDF data sample from DBPedia

```
1  @prefix dbpedia: <http://dbpedia.org/resource/>
2  @prefix dbpprop: <http://dbpedia.org/property/>
3
4  dbpedia:Mona_Lisa   dbpprop:title    "Mona Lisa"@en .
5  dbpedia:Mona_Lisa   dbpprop:title    "La Gioconda"@it .
6  dbpedia:Mona_Lisa   dbpprop:artist   dbpedia:Leonardo_da_Vinci .
7  dbpedia:Mona_Lisa   dbpprop:museum   dbpedia:The_Louvre .
8  dbpedia:The_Louvre  dbpprop:lat      "48.860339"^^xsd:double ;
9                      dbpprop:lng      "2.337599"^^xsd:double ;
10                     dbpprop:director dbpedia:Jean-Luc_Martinez .
```

This syntactic serialization of RDF is named Turtle.[5] The first two lines are the **prefixes**: they can be declared to use a clearer representation of the data, e.g., given the prefix in Line 1, `dbpedia:Mona_Lisa` is a short version of the IRI http://dbpedia.org/resource/Mona_Lisa. The second block (Lines 4–10) contains the RDF statements: each of them is composed of three components, respectively, the subject, the predicate, and the object. The statements in the example describe the Mona Lisa: named Mona Lisa in English (Line 4) and La Gioconda in Italian (Line 5), it is a work of Leonardo da Vinci (Line 6), and it is located at the Louvre (Line 7), a museum located at the coordinates $\langle 48.860339, 2.337599 \rangle$ (Lines 8 and 9) directed by Jean-Luc Martinez (Line 10). Notably, Lines 4–7 clearly show the triples separated by the character "`.`", whereas Lines 8–10 show a convenience syntax that allows to separate with the character "`;`" pairs of property object sharing the same subject.

The following sections illustrate how the RDF data model has been used in the definition of data models for RSP and how it can be processed by the engines. These analyses are used in Sect. 7.3 to present and classify the existing RSP solutions.

7.2.2 The RSP Data Model

In the previous chapters, the notion of data stream was introduced. We learnt that data stream management system (DSMS) (Chap. 5) and complex event processing (CEP) (Chap. 6) systems work on relational data streams. This section, to cope with the data integration problem, presents the **RDF data streams** (or RDF streams), introducing the existing definitions proposed in the previous years. The definition of RDF stream is built from the one of relational data stream. RDF streams are sequences of timestamped data items, where a data item is a self-consumable informative unit; moreover, the main characteristics of relational data streams hold

[5] Cf. http://www.w3.org/TR/turtle/

the following [5]:

- They are **continuous**: new data items are continuously added to the stream.
- They are potentially **unbounded**: a data stream could be infinite.
- They are **transient**: it is not always possible to store data streams in secondary memory.
- They are **ordered**: data items are intrinsically characterized by recency. The order is partial, i.e., two data items can share the same temporal annotation (contemporaneity).

Different definitions of RDF streams have been proposed by RSP research. It is useful to introduce two axes to classify them: the data item (Sect. 7.2.2.1) and the time annotation (Sect. 7.2.2.2).

7.2.2.1 Data Item Dimension

The data item is the minimal informative unit. Existing works in RSP consider two alternatives for this role: RDF statements (triples of RDF resources) and RDF graphs (a set of RDF statements).

The simplest case is the one where the stream is composed of **RDF statements**: each data item is composed of a sequence of three RDF resources—subject, predicate, and object. For example, let us consider the RDF stream in Listing 7.2 in which each triple states the presence of a person in a room of the Louvre:

Listing 7.2 RDF stream

```
:alice   :detectedAt :monaLisaRoom   [1] .
:bob     :detectedAt :parthenonRoom  [2] .
:conan   :detectedAt :monaLisaRoom   [5] .
:bob     :detectedAt :monaLisaRoom   [5] .
```

A Turtle-like syntax is adopted: in each row there is an RDF statement enriched with the time annotation (the fourth element between square brackets). When adopting this data model, each statement may contain enough information to be processed as an informative unit.

Even if the RDF statement stream model is easy to manage, the amount of information that a single RDF statement carries may be not enough when modelling real use cases. For this reason, recent works (e.g., [6]) propose to use as the informative unit **RDF graphs**, i.e., a set of RDF statements. Let us consider the RDF stream presented in Listing 7.3, expressing check-in operations in a social network.

Listing 7.3 RDF stream: social network check-in operation

```
:g1 [1]
  {
    :alice :posts :c1 .
    :c1    :where :monaLisaRoom .
    :c1    :with  :conan .
  }

:g2 [3]
  {
    :bob   :posts :c2 .
    :c2    :where :parthenonRoom .
    :c2    :with  :diana .
  }

:g3 [3]
  {
    :conan :posts :c3 .
    :c3    :where :monaLisaRoom .
  }
```

This example adopts a Trig-like syntax[6]: RDF resources `:g1`, `:g2`, and `:g3` identify three RDF graphs followed by the relative time annotation (the number between squared brackets). The blocks of RDF statements (enclosed in { }) are the contents of the graphs. As it is possible to observe, in this example single RDF statements are not enough to represent a whole informative unit (e.g., a check-in post).

7.2.2.2 Time Annotation Dimension

The time annotation is a set of time instants associated with each data item. The choice to consider the time as an annotation on data items, and not as part of the schema, is inherited by the DSMS research [5], and it is motivated by both modelling and technical reasons: the time information could be part of the schema, but it should not be mandatory (i.e., there are scenarios in which it is not). Moreover, DSMSs and CEPs usually do not allow explicit accesses to the time annotations through the query languages: conditions are usually expressed with time relative constraints, e.g., select events that *happen before a given one*, or identify the events that *hold in the last 5 min*.

The term **application time** refers to the time annotation of a data item [12]. Usually, application time is represented through sets of timestamps, i.e., the identifier of relevant time instants. The classification along the time annotation axis depends on the number of timestamps that compose the application time.

[6] Cf. http://www.w3.org/TR/trig/

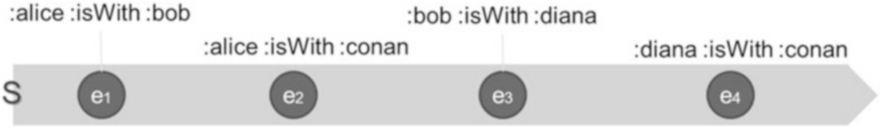


Fig. 7.2 Example of stream with zero timestamps per data item

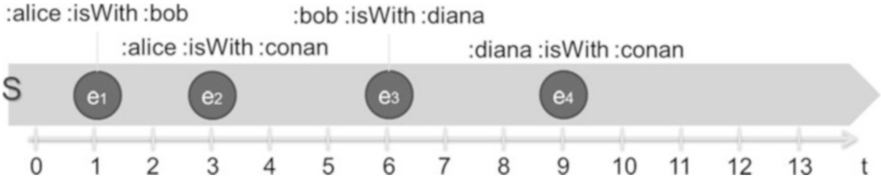


Fig. 7.3 Example of stream with one timestamp per data item

In the simplest case, the application time consists of **zero timestamps**: in other words, there is no explicit time information associated with the data items. It follows that the RDF stream is an ordered sequence of elements that arrive at the processing engine over time, like in the stream *S* represented in Fig. 7.2.

Rounds labeled with e_i (i in $[1,4]$) represent the data items of the stream; the time flows from left to right, e.g., item e_1 takes place before item e_2. Even if the data items do not have explicit timestamps, it is possible to process those streams by defining queries that exploit the order of the elements, such as follows:

1. Does Alice meet Bob before Conan?
2. Who does Conan meet first?

Let us consider now the case in which the application time is modelled introducing a metric [35] and each data item has **one timestamp** like in Fig. 7.3.

In most of the existing works, the timestamp used in the application time represents the time instant at which the associated event occurs, but other semantics are possible, e.g., time since the event holds. Due to the fact that data items are still ordered by recency (as in the previous case), it is possible to issue queries of the previous case, as q_1 and q_2. Additionally, it is possible to write queries that take into account the time, such as follows:

1. Does Diana meet Bob and then Conan within 5 min?
2. How many people has Alice met in the last 5 min?

It is worth noting that q_3 and q_4 do not refer to absolute time instants but to relative ones with respect to the time instant of another event (as in q_3) or the current time instant (as in q_4).

As a final case, let us introduce the application time composed of **two timestamps**. The semantics that is usually associated with the two timestamps is the time range (*from*, *to*] in which the data item is valid, as shown in Fig. 7.4.

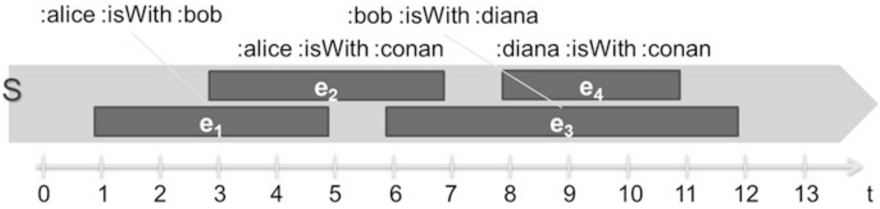


Fig. 7.4 Example of stream with two timestamps per data item

Each square represents a data item, and the application time is represented by the initial and the final timestamps, e.g., e_1 has application time $(1, 5)$, so it is valid from the time instant 1 to the time instant 5. Similar to the previous case, it is still possible to process the queries presented in the first two cases (e.g., $q_1, \ldots, q_4$), and additionally more complex constraints can now be written, such as follows:

1. Which are the meetings that last less than 5 min?
2. Which are the meetings with conflicts?

Other cases exist, where the application time is composed of three or more timestamps or where the application time semantics has other meanings. Even if they can be useful in some use cases, they are still under investigation in RSP research and no relevant results have been reached, yet.

7.2.3 RSP Query Model

After the presentation of the RDF stream definitions, this section discusses the problem of processing those kinds of data. As for the data model, the RSP query model is inspired by research in DSMS and CEP: existing works adapt the existing operators of those disciplines to process RDF data streams.

7.2.3.1 The CQL Model and Its RDF Stream Adaptation

In Chap. 5, we learnt that the continuous query language (CQL) stream processing model (proposed by the InfoLab of Stanford University [4]) defines a generic DSMS through three classes of operators (Fig. 7.5a).

The first class of operators manages the data stream: due to the fact that a stream is a potentially infinite bag of timestamped data items, these operators extract finite bags of data. CQL defines as **stream-to-relation** the operators that are able to transform streams in relations; among the available operators of this class, one of the most studied is the sliding window. Next, the relation may be transformed into another relation through a **relation-to-relation** operator. Relational algebraic expressions are well-known cases of this class of operators. Finally, the transformed

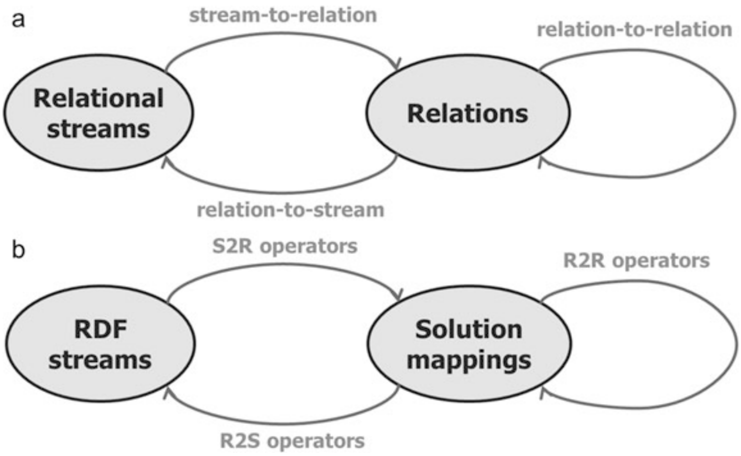


Fig. 7.5 The CQL model (**a**) and its adaptation for RDF Stream Processing (**b**)

relation has to be set as output. There are usually two ways to set the answer: as a time-varying relation and as a part of a stream. In the second case, an additional operator is required: the **relation-to-stream** one.

As already mentioned in Chap. 5, the CQL model inspired the design of different RDF Stream Processing engines [10, 14, 26], and currently there are different implementations (e.g., C-SPARQL, SPARQLstream, CQELS) of the adapted model, represented in Fig. 7.5b.

The stream and the relation concepts are mapped to RDF streams and to a set of mappings (using the SPARQL algebra terminology[7]), respectively. To highlight the similarity of the RSP operators to the CQL ones, similar names are used: **S2R**, **R2R**, and **R2S**, to indicate the operators, respectively, analogous to stream-to-relation, relation-to-relation, and relation-to-stream operators. The following sections describe the sliding window as an example of S2R operators, SPARQL algebra as an example of R2R operators, and the streaming operator as an instance of R2S operators.

7.2.3.2 The Sliding Windows

Given a stream *S*, a sliding window dynamically selects subsets of the data items of *S*. The intuition behind this operator is that the most recent data are the most relevant, so it selects them from the stream and queries it, repeating these operations as time goes by.

[7] Cf. http://www.w3.org/TR/sparql11-query/

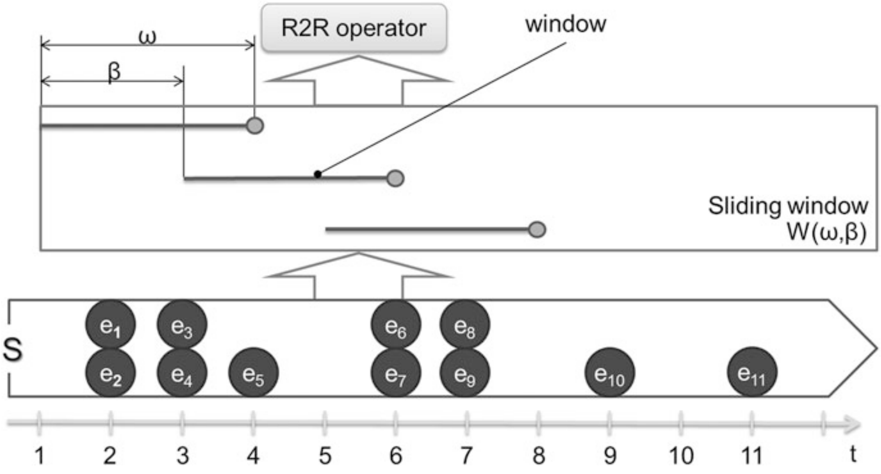


Fig. 7.6 Time-based sliding window

The basic elements of the sliding windows are the **windows**. Given a stream S, a **time-based window** W defined through two parameters o and c selects the data item d of S with associated timestamp in $(o, c]$. Given the stream in Fig. 7.6, the window defined between the time instants 2 (included) and 5 (excluded) selects the elements e_3, e_4, and e_5.

A sliding window generates multiple windows (at different time instants) to create a time-varying view over the stream. Two of the most famous kinds of sliding windows are the **time-based sliding windows** and the **triple-based sliding windows**. A time-based sliding window generates a sequence of windows at regular time intervals (e.g., a window each 2 s). Then, it selects the contents of the streams according to each window $(o, c]$ (where o and c are the opening and the closing time instants). The time range changes periodically, modifying the content of the sliding window. A time-based sliding window is described through two parameters, the **width** ω (the dimension of the window, $c - o$) and the **slide** β (the distance between two consecutive windows).

Let us consider the example in Fig. 7.6: it shows a time-based window W with width $\omega = 3$ and slide $\beta = 2$. The first window selects all the items with timestamps in (1,4], the second one selects the items with timestamps in (3,6], and the third one selects the items with timestamps in (5,8]. As mentioned above, the width of the window represented in the picture is 3 ($4 - 1 = 6 - 3 = 8 - 5$), while the slide parameter, defined as the distance between two consecutive windows, is 2 ($3 - 1 = 5 - 3$). When the slide parameter is equal to the width parameter, the sliding window is also known as **tumbling window**, as depicted in Fig. 7.7. This kind of sliding window is important because it partitions the stream: each data item would only be in one window.

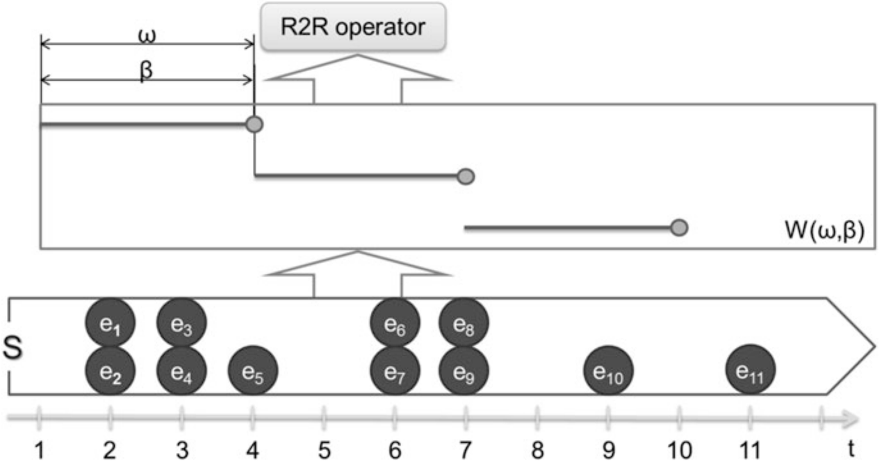


Fig. 7.7 Time-based tumbling window

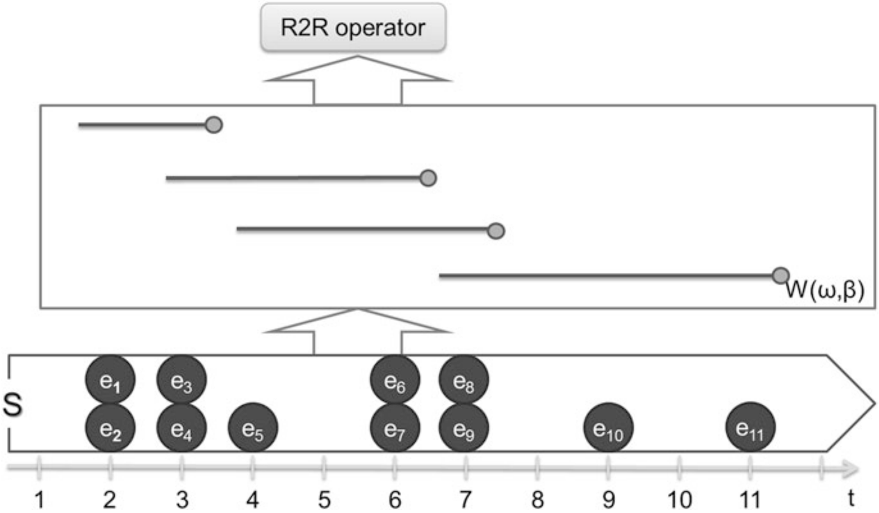


Fig. 7.8 Tuple-based tumbling window

Tuple-based sliding windows select a fixed number of data items (Fig. 7.8). Similar to time-based sliding windows, they are described by the width and the slide parameters, but the width indicates the number of RDF statements that are collected in the current view, while the slide indicates how many RDF statements are removed/added at each window slide. Consequently, the difference between the closing time instant and the opening time instant of the generated windows is not constant: as can be observed in the picture, windows can have different lengths.

7.2.3.3 The R2R Operator

SPARQL is the query language of RDF [33]: it allows to query RDF models and to perform relation transformations. In the following, the main concepts of the SPARQL algebra [32] are briefly introduced, focusing on the `WHERE` clause, the clause that contains the criteria of selection of the data (similar to the `WHERE` clause of SQL). In SPARQL, the `WHERE` clause contains a set of **graph pattern** expressions that can be constructed using the operators `OPTIONAL`, `UNION`, and `FILTER` and concatenated via a point symbol "`.`" that means `AND`. Let us extend the sets of symbols introduced for RDF [i.e., the pairwise disjoint sets I (IRIs), B (blank nodes), and L (literals)] with the set V (variables). A **graph pattern expression** is defined recursively as follows:

1. A tuple from $(I \cup B \cup V) \times (I \cup V) \times (I \cup B \cup L \cup V)$ is a graph pattern and, in particular, it is a triple pattern.
2. If P_1 and P_2 are graph patterns, then $(P_1 \,.\, P_2)$, $(P_1$ OPTIONAL $P_2)$, and $(P_1$ UNION $P_2)$ are graph patterns.
3. If P is a graph pattern and R is a SPARQL built-in condition, then $(P$ FILTER $R)$ is a graph pattern.

A SPARQL built-in condition is composed of elements of the set $I \cup L \cup V$ and constants; logical connectives ($\neg$, $\wedge$, $\vee$); ordering symbols ($<, \leq, \geq, >$); the equality symbol ($=$); unary predicates like `bound`, `isBlank`, and `isIRI`; and other features. An important case of graph pattern expression is the **basic graph pattern** (BGP): it is defined as a set of triple patterns and `FILTER` clauses that are connected by the "`.`" (i.e., the `AND`) operator.

The semantics of SPARQL queries uses as the basic building block the **solution mapping**: let P be a graph pattern; $var(P)$ denotes the set of variables occurring in P. A **solution mapping** μ is a partial function $\mu : V \rightarrow (I \cup L \cup B)$. The domain of μ, denoted by $dom(\mu)$, is the subset of V where μ is defined.

The relation between solution mappings, triple patterns, and basic graph patterns is given in the following definition: given a triple pattern t and a solution mapping μ such that $var(t) \subseteq dom(\mu)$, $\mu(t)$ is the triple obtained by replacing the variables in t according to μ. Given a basic graph pattern B and a solution mapping μ such that $var(B) \subseteq dom(\mu)$, we have that $\mu(B) = \cup_{t \in B}\mu(t)$, i.e., $\mu(B)$ is the set of triples obtained by replacing the variables in the triples of B according to μ.

Using these definitions, Pérez et al. [32] define the semantics of SPARQL queries as an algebra. The main algebra operators are join ($\bowtie$), union ($\cup$), difference ($\setminus$), and left join ($⟕$). The authors define the semantics of these operators on **sets of solution mappings** denoted by Ω. The evaluation of a SPARQL query is based on its translation into an algebraic tree composed of these algebraic operators.

The simplest case is the **evaluation of a basic graph pattern**: let G be an RDF graph over $I \cup L$ and P a basic graph pattern. The evaluation of P over G, denoted by $[\![P]\!]_G$, is defined by the set of solution mappings:

$$[\![\mathrm{P}]\!]_G = \{\mu |\ dom(\mu) = var(P) \text{ and } \mu(P) \subseteq G\}$$

If $\mu \in [\![P]\!]_G$, μ is said to be a solution for P in G.

The evaluation of more a complex graph pattern is compositional and can be defined recursively from the basic graph pattern evaluation.

7.2.3.4 The Streaming Operators

When the R2R operator transforms a set of mapping, the system can produce the output with the computation result. If the output of the query processor should be a stream, it is necessary to include an R2S operator. When applied, it appends the set of mappings to the output stream. At each time instant at which the continuous query is evaluated, the set of solution mappings is processed by the R2S operator, which is in charge in determining which data items have to be streamed out. There are usually three R2S operators, depicted in Fig. 7.9.

Rstream streams out the computed timestamped set of mappings at each step. Rstream answers can be verbose as the same solution mapping can be computed at different evaluation times and consequently streamed out. It is suitable when it is important to have the whole SPARQL query answer at each step, e.g., discover museum attractions that are popular in a social network in the recent time period.

Istream streams out the difference between the timestamped set of mappings computed at the last step and the one computed at the previous step. Answers are usually short (they contain only the differences), and consequently this operator is used when data exchange is expensive. Istream is useful when the focus is on the new mappings that are computed by the system, e.g., discover emerging museum attractions in a social network.

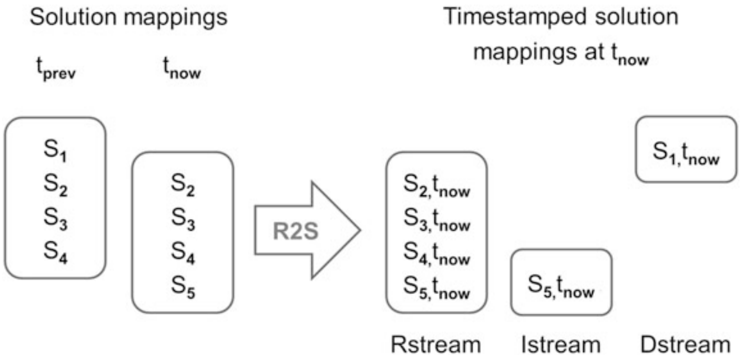


Fig. 7.9 Streaming operators

Dstream does the opposite of Istream: it streams out the difference between the computed timestamped sets of mappings at the previous step and at the last step. Dstream is normally considered less relevant than Rstream and Istream, but it can be useful, for example, to retrieve attractions that are no more popular.

7.2.3.5 Temporal Operators

So far only typical operators of DSMS were introduced. They allow to match graph patterns in the streaming data. However, practical scenarios (including the one illustrated in Sect. 7.4) may require to check if two graph patterns are observed at the same time, one after the other, or when typical temporal relations between intervals hold.

For instance, given the situation illustrated in Fig. 7.10, one may be willing to detect the people who are now with people who were observed together. At time 6, the answer is Conan and Diana, because they are with Alice and Bob who were observed together between 1 and 5.

Figure 7.11 informally introduces the temporal relations between two temporal intervals as defined in Allen's algebra [2, 35]. Assume that compatible solution

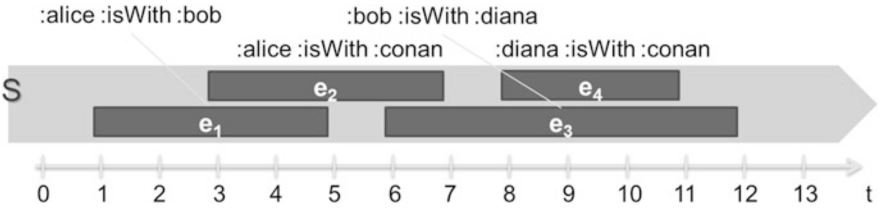


Fig. 7.10 Example of stream with two timestamps per data item

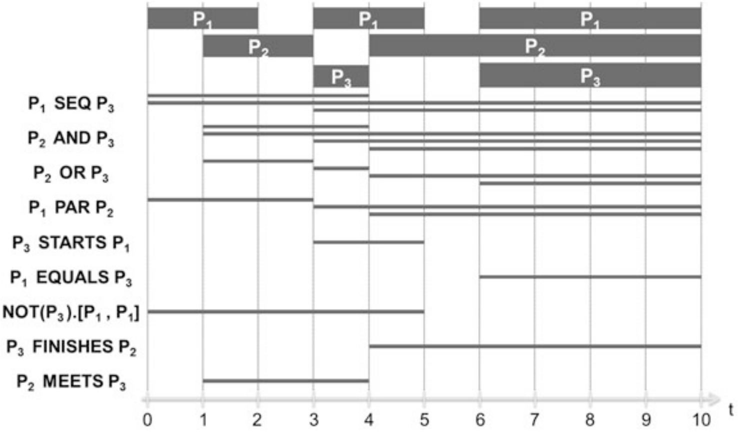


Fig. 7.11 Temporal relations between two temporal intervals as defined in Allen's algebra

mappings exist for three graph patterns P_1, P_2, and P_3 in the time intervals shown in Fig. 7.11; the horizontal bars represent the result of evaluating the operators on the solution mappings at different time units depicted with vertical dashed lines.

A subset of those temporal operators can be added to the RSP query model as special types of joins that add to the Boolean semantics of the join operator, the temporal semantics of the specific temporal relation of the operator. For instance, the solution mappings of P_1 `SEQ` P_2 from a membership perspective are those of (P_1 `JOIN` P_2), but the `SEQ` operator keeps only the solution mappings of the `JOIN` operator for which a solution mapping of P_2 starts after the end of a solution mapping of P_1.

7.3 RSP Implementations

The previous section explains the data models and the query models used by RDF Stream Processing systems: putting together techniques and concepts from DSMS, CEP, and semantic Web researches, RSP engines are systems able to cope with the semantic heterogeneity of the data. This section presents an overview of existing systems.

Table 7.1 lists the systems and classifies them according to the RSP data and query models presented above. Analyzing the table, it is possible to observe that most systems focus on RDF streams with RDF statement as the data item and the application time is composed of one timestamp. C-SPARQL [10], CQELS [26], and SPARQL$_{stream}$ [14] manage data streams where data items are RDF statements. Their query models are similar, but they are designed in different ways and target different use cases. Sections 7.3.1–7.3.3 discuss them.

Table 7.1 Data models adopted by RSP systems

	C-SPARQL	CQELS	SPARQL$_{stream}$	INSTANS	ETALIS	SLD
Data item	Statement	Statement	Statement	Statement	Statement	Graph
Application time	1	1	1	0	2	1
Time-based sliding windows	✓	✓	✓			✓
Triple-based sliding windows	✓	✓				✓
S2R	RStream	IStream	RStream IStream DStream	RStream	RStream	RStream
SEQ					✓	
EQUALS					✓	

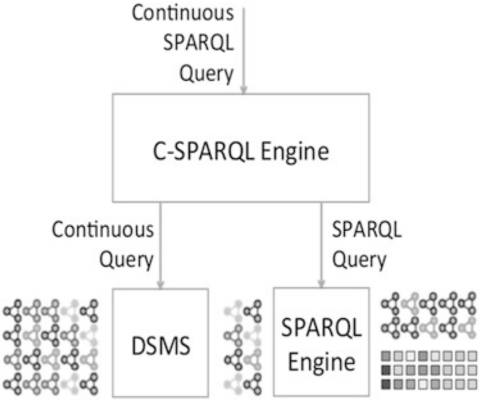
The data model with one timestamped RDF statement is the one on which the initial RSP research focused, and novel trends started to consider data model variants. The Streaming Linked Data (SLD) platform, described in Sect. 7.3.6, has features similar to C-SPARQL, CQELS, and SPARQL$_{stream}$, but it is able to process RDF streams with RDF graphs as data items. INSTANS (Sect. 7.3.4) follows a completely different approach: it takes sequences of RDF statements without timestamps as input and processes them through the RETE algorithm. Finally, ETALIS Language for Events (ETALIS) and Event Processing SPARQL (EP-SPARQL), presented in Sect. 7.3.5, work on RDF streams with application time composed of two timestamps: they use CEP concepts and are able to perform Stream Reasoning tasks.

7.3.1 C-SPARQL

Continuous SPARQL (C-SPARQL) [10] is a language for continuous queries over streams of RDF data that extends SPARQL 1.1. It is implemented in the C-SPARQL engine, and it allows to register queries that are continuously evaluated over time.

Figure 7.12 presents a high-level description of the C-SPARQL engine architecture. The engine is capable of registering queries and running them continuously, according to the configuration of the engine. To this end, the C-SPARQL engine uses two subcomponents, a data stream management system and a SPARQL engine. The former is responsible for executing continuous queries over RDF streams, producing a sequence of RDF graphs over time, while the latter runs a standard SPARQL query against each RDF graph in the sequence, producing a continuous result. This result is finally formatted as specified in the query: if the SELECT or ASK form is used, then the result is a relational data stream; if the CONSTRUCT form is used, the result is an RDF stream. Both the SPARQL engine and the DSMS are plug-ins of

Fig. 7.12 Architecture of the C-SPARQL engine



the C-SPARQL engine, and they can be changed. The binaries of the engines use Esper[8] as the DSMS and Apache Jena-ARQ[9] as the query engine.

7.3.2 CQELS

The Continuous Query Evaluation over Linked Streams (CQELS) accepts queries in CQELS-QL [26]—a declarative query language built from SPARQL 1.1 grammar. Like C-SPARQL, it extends SPARQL with operators to query streams. The main difference between C-SPARQL and CQELS-QL is in the R2S operator supported: CQELS-QL supports only Istream, whereas C-SPARQL supports only Rstream.

Different from the C-SPARQL engine that uses a "black box" approach which delegates the processing to other engines, CQELS proposes a "white box" approach (see Fig. 7.13) and implements the required query operators natively to avoid the overhead and limitations of closed system regimes.

CQELS provides a flexible query execution framework with the query processor dynamically adapting to the changes in the input data. During query execution, it continuously reorders operators according to heuristics that improve query execution in terms of delay and complexity. Moreover, external disk access on large linked data collections is reduced with the use of data encoding and caching of intermediate query results. It returns both data streams and RDF streams depending on the query form used.

7.3.3 SPARQL$_{stream}$

SPARQL$_{stream}$ [14] is another extension of SPARQL that supports operators over RDF streams such as time windows. Unlike CQELS and the C-SPARQL engine,

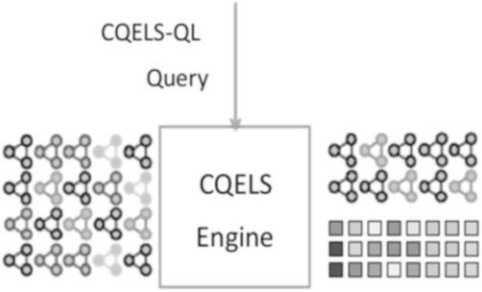


Fig. 7.13 Architecture of CQELS

[8] Cf. http://esper.codehaus.org/

[9] Cf. http://jena.apache.org/documentation/query/

SPARQL$_{stream}$ supports all the streaming operators presented in Sect. 7.2.3.4. The language is adopted in morph-streams: it is an RDF stream query processor that uses ontology-based data access techniques [15] for the continuous execution of SPARQL$_{stream}$ queries against virtual RDF streams that logically represent relational data streams.

Morph-streams use R2RML[10] to define mappings between ontologies and data streams. SPARQL$_{stream}$ queries are rewritten in continuous queries over the data streams. The rewriting process does not translate directly a SPARQL$_{stream}$ query in a continuous query in the target language of the underlying DSMS (Fig. 7.14). It represents, instead, the query as a relational algebra expression extended with time window constructs. This allows performing logical optimizations (including pushing down projections, selections, and join and union distribution) and translating the algebraic representation into a target language or REST API request.

The algebraic representation can be translated into both DSMS continuous queries, e.g., SNEE[11] or Esper, and sensor middleware REST API invocation, e.g., GSN[12] or Cosm/Xively.[13]

7.3.4 INSTANS

INSTANS (Incremental eNgine for STANding Sparql) [34] takes a different perspective on RDF Stream Processing. It asks the users to model their continuous

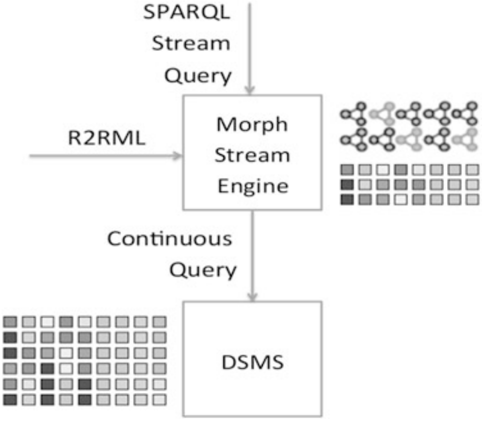


Fig. 7.14 Architecture of morph-streams

[10] Cf. http://www.w3.org/TR/r2rml/

[11] Cf. http://code.google.com/p/snee/

[12] Cf. http://lsir.epfl.ch/research/current/gsn/

[13] Cf. https://xively.com/

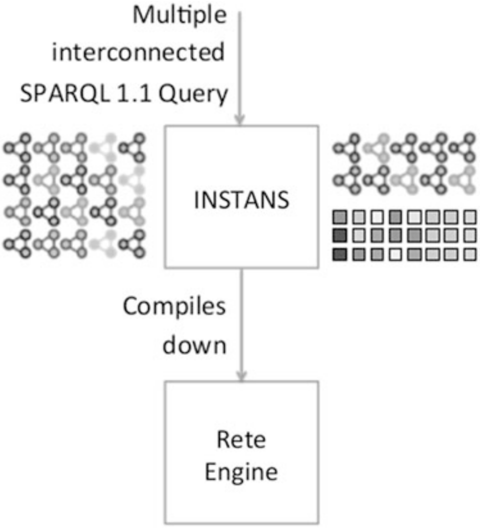


Fig. 7.15 Architecture of INSTANS

query problem as multiple interconnected SPARQL 1.1 queries and rules. It performs continuous evaluation of incoming RDF data against the compiled set of queries, storing intermediate results. It has no notion of S2R operator; when all the conditions of a query are matched, the result is instantly available. For this reason, it requires no extensions to RDF or SPARQL. As shown in Fig. 7.15, to process the multiple interconnected queries, it compiles them into a RETE-like structure.

7.3.5 ETALIS and EP-SPARQL

ETALIS (Event TrAnsaction Logic Inference System) [3] is a CEP with Stream Reasoning features. As the input data format, it uses RDF statements annotated with two timestamps (see also Sect. 7.2.2). Users can specify event processing tasks in ETALIS using two declarative rule-based languages called ETALIS Language for Events (ELE) and Event Processing SPARQL (EP-SPARQL) [3] . Both languages have the same semantics: complex events are derived from simpler events by means of deductive prolog rules (Fig. 7.16).

ETALIS not only supports typical event processing constructs (e.g., sequence, concurrent conjunction, disjunction, and negation), but it also supports reasoning about events. For example, as CEPs, it allows to check for sequences such as $A \rightarrow B$ (an event of type A followed by an event of type B).

However, it also allows stating that C is a subclass of A. Therefore, the condition $A \rightarrow B$ will be matched also if an event of type C is followed by an event of type B, because all events of type C are also events of type A.

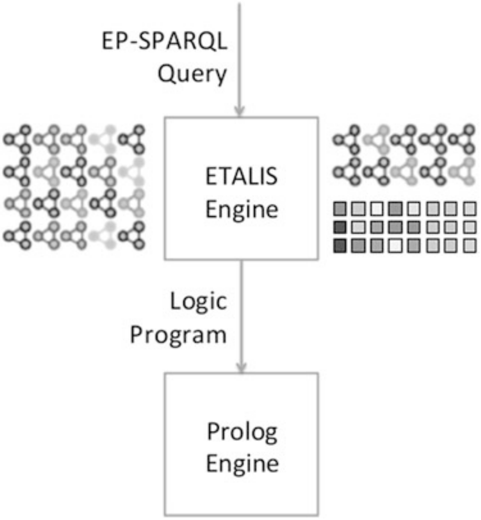


Fig. 7.16 Architecture of ETALIS

Moreover, the two languages support all operators from Allen's interval algebra (e.g., during, meets, starts, finishes, as described in Sect. 7.2.3.5); count-based sliding windows; event aggregation for `COUNT`, `AVG`, `SUM`, `MIN`, and `MAX`; event filtering, enrichment, projection, translation, and multiplication; processing of out-of-order events; and event retraction (revision). ETALIS is a pluggable system that can use multiple prolog engines such as YAP,[14] SWI,[15] SICStus,[16] XSB,[17] tuProlog[18], and LPA Prolog.[19]

7.3.6 SLD

SLD is not a proper RSP engine, but it wraps the C-SPARQL engine in order to add support features, such as extensible means for real-time data collection, linked data publishing, and data and query result visualizing. As depicted in Fig. 7.17, the SLD framework offers a set of adapters that transcode relational data streams in streams of RDF graphs (e.g., a stream of micro-posts as an RDF stream using

[14] Cf. http://www.dcc.fc.up.pt/~Evsc/Yap/

[15] Cf. http://swi-prolog.org/

[16] Cf. http://www.sics.se/isl/sicstuswww/site/index.html

[17] Cf. http://xsb.sourceforge.net/

[18] Cf. http://www.alice.unibo.it/xwiki/bin/view/Tuprolo

[19] Cf. http://www.lpa.co.uk/win.html

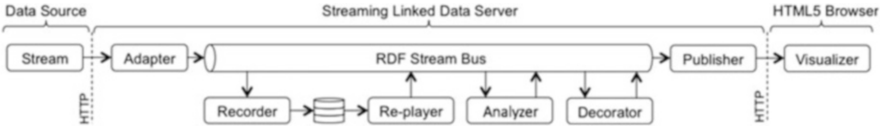


Fig. 7.17 Architecture of SLD

the SIOC vocabulary [13] or a stream of weather sensor observation using the Semantic Sensor Network vocabulary [17]), a publish/subscribe bus to internally transmit RDF streams, facilities to record and replay RDF streams, an extendable component to decorate RDF streams (e.g., adding sentiment annotations to microposts), and a linked data server to publish results following the Streaming Linked Data Format [8].

SLD is realized in order to simplify the task of deploying the C-SPARQL engine in real-world applications: it is at the basis of works of real-time data analysis of city-scale events (i.e., a group of events located in multiple venues around a city) for London Olympic Games 2012, Milano Design Week 2013, and Milano Design Week 2014 [6].

7.4 Case Study

This section puts at work, in a realistic cultural heritage case study, the elements introduced in the previous sections. It proposes Blitz Guided Thematic Tours as an engaging cyber-physical-social service to explore a museum. This service allows free guides for offering a short and focused guided tour that matches the interests of the people nearby with the artworks exposed in the rooms at hand and with their own expertise.

The section first describes the requirements such a service has to implement. Then, it presents an overview of an RSP-based implementation. The emphasis is posed on the data (both streaming and static) and on the continuous queries that process it.

The service needs to address the following **requirements**:

- R.1 It must be able to merge information from multiple heterogeneous data streams.
- R.2 It must allow to detect where the visitors are in the museum with a minimum delay.
- R.3 It should understand what the visitors are interested in.
- R.4 It should recommend free guides to organize a thematic tour.

The **data** involved in the Blitz Guided Thematic Tour is collected in different ways and from different data sources. It is worth noting that the visitors are not only service consumers but also data producers. The means to collect the data are as follows:

- User Mobile App: the visitors, when entering the museum, are offered to download the official app of the museum. This app offers information about artworks using NFC technology and allow guides to invite visitors to join a Blitz Guided Thematic Tour.
- Guide Mobile App: the guides also have a mobile app that lets them specify their status (e.g., if they are available to start a tour) and their position. This application represents the communication channel between the guides and the visitors involved in a Blitz Guided Thematic Tour.
- NFC sensors: these sensors are placed beside every artwork. They identify the artwork and allow visitors to learn more about the artwork. In addition, they allow indoor positioning of the visitors.
- Social networks: they are the only sources of knowledge about the users. The official User Mobile App simplifies visitors' task of sharing online the artworks they liked by asking visitors to connect their social network accounts. For the users who do so, the service can learn the language(s) and the interests of the users analyzing past micro-posts (e.g., using the technique described in [7]).
- Static knowledge: information related to the artworks, the rooms they are displayed in, how the rooms are connected, and the expertise of the guides.

Figure 7.18 presents the data model of the case study. `Users` (i.e., `Visitors` and `Guides`) are identified in the system via the Universally Unique Identifier (UUID) of their `Smartphone` where the two official mobile applications are installed: the part with dot-dash line of the figure presents the data related to each `Visitor`, in particular the information the service can obtain from the social network; the dotted part refers to the `Guides`' information, their positions and their statuses; finally, the dashed part models the static information about each `NFC Sensor` that `isAssociatedWith` an `ArtWork` which `isLocatedIn` a `Room` which `isConnectedTo` one or more `Rooms`.

Figure 7.19 offers an overview of the architecture of the Blitz Guided Thematic Tour service with a focus on data streams, static data, and the components that

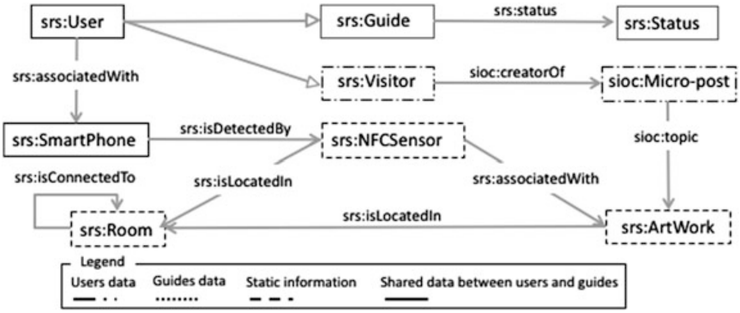


Fig. 7.18 Data model of the Blitz Guided Thematic Tour case study

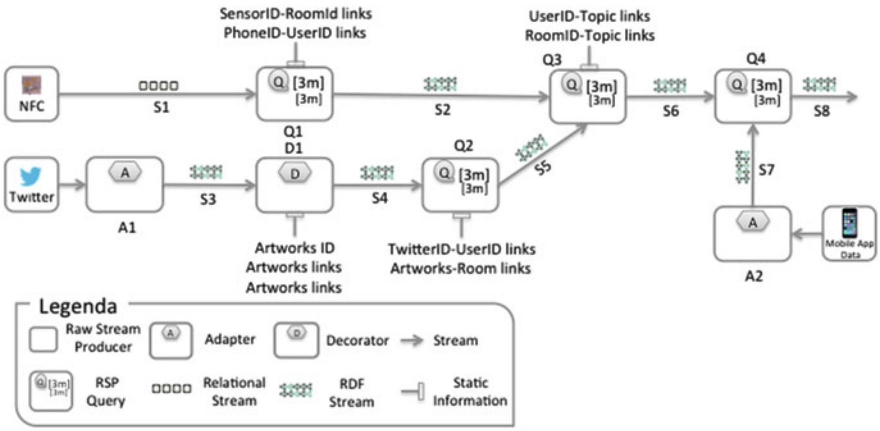


Fig. 7.19 Overview of the architecture of the Blitz Guided Thematic Tour service

transform the information into actionable knowledge (i.e., recommendations for free guides). The visual language is the one proposed for the Streaming Linked Data framework [6]. Each component represents a step in the data transformation process, while the input and output data are depicted in a different way depending on whether they are static or streaming. The remaining part of this section details this picture step by step referring to the numbering of the elements.

Let us start from the NFC sensor. Smartphones equipped with NFC represent the cheapest way for indoor positioning. In the case study, the visitors' positions are acquired as a result of the action of learning more about artworks exhibited in the museum. The NFC reader produces a relational stream (see component S1 in Fig. 7.19) presented in Listing 7.4 containing the tuples `<phoneID,sensorID, timestamp>`.

Listing 7.4 Two events of stream S1 produced by NFC sensors and consumed by query Q1. For simplicity we omit timestamps in the following listings

```
phone1, monnalisaSensor [$\tau_1$]
phone2, theKissSensor [$\tau_2$]
```

Query Q1 addresses the requirement R.2. It continuously processes this relational stream treating it as a *virtual RDF stream* by using the Ontology-Based Data Access (OBDA) approach briefly introduced in Sect. 7.1. A mapping file, written in R2RML, defines mappings between the ontology illustrated in Fig. 7.18 and the relational data streams. It maps the relational data in the streams to the objects in the ontology. It is worth noting that in order to save space, the prefix clauses are shown only the first time the prefix is required, and they are omitted in all the listings that appear afterward.

Listing 7.5 contains the R2RML file to map the detection of a smartphone by a sensor. Line 9 declares how to produce the subject of the triple using the

Listing 7.5 R2RML file to map the detection of a smartphone by a sensor

```
 1  @prefix rr: <http://www.w3.org/ns/r2rml#> .
 2  @prefix srs:<http://sr.org/ontologies/2014/7/srs#> .
 3  @prefix : <http://sr.org/R2RMapping#> .
 4  @prefix phone:<http://sr.org/data/smartphones/>
 5  @prefic sensor:<http://sr.org/data/sensors/>
 6
 7  :smartphoneMap a rr:TriplesMap ;
 8    rr:logicalTable [ rr:tableName "S1" ];
 9    rr:subjectMap [ rr:template "phone:{smartphoneID}" ];
10    rr:predicateObjectMap [
11      rr:predicate srs:detectedBy;
12      rr:objectMap [ rr:template "sensor:{sensorID}" ] ] ].
```

`phone` prefix along with the `smartphoneID` parameter extracted from the stream `S1` specified at Line 8. The predicate of the triples, `srs:detectedBy`, is created at Line 10 along with the object, specified at Line 12 using the template URL for the NFC sensors.

Listing 7.6 presents the query that processes the relational data stream S1 treating it as a virtual RDF stream. It is worth noting that we are in a streaming setting; the queries are registered and periodically run against the flowing data. The `REGISTER STREAM` clause at Line 1 denotes the registration process for the query and tells the system to produce a stream as output. It must be coupled with the `CONSTRUCT` clause, specified at Line 3, to tell the system the format of the output stream. The `FROM STREAM` clause at Line 4 specifies the query input stream. The unbound nature of a stream requires the ability to specify certain criteria to select a part of flowing data (e.g., a time-based window sorting out the most recent triples). The `RANGE` and `STEP` clauses identify the time-based sliding window containing the data to analyze; at Line 3 we specified a 10 min width window with a slide of 1 min. The `FROM` clause identifies the online source of RDF static data. The `WHERE` clause at Line 5 opens the triple pattern specifications. The triple pattern at Line 5 matches all the triples with the `srs:detectedBy` predicate from the current window over the stream to find the links between smartphones and sensors. Then, the triple patterns at Line 6 and Line 7 are matched against the static knowledge base, to identify the users associated with the smartphones (`?userID`) and the room in which a sensor is located (`?roomID`).

Listing 7.6 Query Q1 that processes stream S1 and produces stream S2

```
1  REGISTER QUERY Q1 AS
2  CONSTRUCT { ?userID srs:detectedAt ?roomID }
3  FROM STREAM <http://sr.org/streams/S1> [RANGE 10m STEP 1m]
4  FROM <http://sr.org/staticData/museum.rdf>
5  WHERE {  ?phoneID srs:detectedBy ?sensorID ;
6                    srs:associatedWith ?userID .
7           ?sensorID srs:locatedIn ?roomID . }
```

Let us assume that `user:123` and `user:345`, respectively, own `phone1` and `phone2`. Query Q1, observing as input the relational data stream in Listing 7.4, produces as output on the RDF stream S2 the triples shown in Listing 7.7.

Listing 7.7 An event in stream S2 produced by query Q1 and consumed by query Q3

```
@prefix rooms:<http://sr.org/data/rooms/>
@prefix topics:<http://sr.org/data/topics/>
@prefix users:<http://www.sr.org/data/users/>

users:123 srs:detectedAt  rooms:31 .
users:345 srs:detectedAt  rooms:42 .
```

Let us now focus on the application installed on the visitor's smartphone. It represents the communication channel between the museum and the users. Once installed, the application, via a configuration wizard, asks the users to insert their social network account credentials (e.g., Twitter). Using Twitter credentials, a Twitter adapter (see component A1 in Fig. 7.19) creates a new RDF stream (S3) *wrapping* the Twitter stream API. The information about the micro-posts of the users flows on this RDF stream. Listing 7.8 shows an example of a micro-post in RDF that flows on this stream. The vocabulary used to describe micro-posts is an extension of the Semantically-Interlinked Online Communities (SIOC) ontology [13]. It states that the micro-post `@Louvre #monnalisa #wow` is posted by `user:543`, it contains the hashtags `monnalisa` and `wow`, and the smartphone was located in a given geo-position.

Listing 7.8 An event in stream S3 produced by adapter A1 and consumed by decorator D1

```
@prefix geo:<http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix sioc:<http://rdfs.org/sioc/ns#> .
@prefix mp:<http://www.sr.org/data/mp/>
@prefix tags:<http://www.sr.org/data/tags/>

mp:1234 sioc:content  "@Louvre #monnalisa #wow"^^xsd:string ;
  sioc:has_creator  user:543 ;
  sioc:topic tag:monnalisa, tag:wow ;
  geo:location [ a geo:SpatialThing ;
     geo:lat   "48.860816"^^xsd:double ;
     geo:long  "2.338320"^^xsd:double ] .
```

The Twitter decorator (D1) enriches stream S3 by semantically annotating the micro-post that flows on the stream with links that point to the artworks in the static knowledge. For instance, it adds the following triple to the micro-post in Listing 7.8:

```
  mp:123456789 sioc:topic <http://sr.org/data/artworks/
Monnalisa>
```

Several techniques can be employed to discover these links. For instance, the Streaming Linked Data framework [7] uses an aggregation of a spatial metric (which checks the geo-position of the micro-post to make sure it was posted in the neighbourhood of the museum) and four lexical similarity metrics that compare the content and hashtags of each tweet to the names of the artworks and the bags of words that describe them.

Query Q2 addresses the requirement R.2 (i.e., detecting the position of the users) using the stream of tweets to position the visitors. Notably, Q2 complements the positions obtained with the readings of the NFC sensor (see Q1). It does so by processing stream S4 and positioning the users in the rooms based on the artworks they have been talking about in the last 10 min. This continuous query operates on a *native RDF stream*; thus, it does not need a mapping.

Listing 7.9 Query Q2 that processes stream S4 and produces stream S5

```
REGISTER QUERY Q2 AS
PREFIX srs:<http://streamreasoning.org/ontologies/2014/7/srs#>
PREFIX sioc:<http://rdfs.org/sioc/ns#>
CONSTRUCT { ?userID srs:detectedAt ?roomID }
FROM STREAM <http://sr.org/streams/S4> [RANGE 10m STEP 1m]
FROM <http://sr.org/staticData/museum.rdf>
WHERE {
   ?mp sioc:has_creator ?userID ;
   sioc:topic ?artworksID .
   ?artworksID srs:isIn ?roomID .
}
```

Using the decorated micro-posts that flow on stream S4, e.g., the one illustrated in Listing 7.8, query Q2, presented in Listing 7.9, links the user to the room where the artwork is exhibited. On its output stream S5, it pushes triples such as follows:
`users:543 srs:detectedAt rooms:31`

It is worth noting that the output produced by queries Q1 and Q2 is made of triples of the same type. This is typical of OBDA systems where complementary (and sometimes even overlapping) information is obtained from multiple data sources.

Query Q3, presented in Listing 7.10, answers R.1 (i.e., merging information from multiple heterogeneous streams) and R.3 (i.e., understanding what the visitors may be interested in). For each room, it counts the number of visitors that may be interested in a given topic. It outputs the top five most the largest groups (see `ORDER BY` and `LIMIT` clauses at Lines 18 and 19) formed by at least ten visitors (see `HAVING` clause at Line 17). It does so by consuming the RDF streams of visitors' positions produced by Q1 and Q2 (see the two different `FROM STREAM` clauses at Lines 4 and 5) and some static data describing the topology of the museum and the topics that may interest the visitors. The former is used to collect the visitors and the artworks in a given room or in a room nearby, while the latter is computed using an off-line analysis of the tweets of the visitors (e.g., using the approach described in [7]), and it is used to match topics of interest to topics describing the artworks.

Notably, the match requires to reason on the relations between the topics of the artworks and the topics of interest of the visitors. The query does so by considering that the property `skos:transitiveBroader` (see Line 14) is transitive and by traversing the graph that links the two topics in order to find a matching.

Listing 7.10 Query Q3 that processes streams S2 and S5 and produces stream S6

```
 1  REGISTER QUERY Q3 AS
 2  PREFIX srs:<http://streamreasoning.org/ontologies/2014/7/srs#>
 3  CONSTRUCT { ?roomID srs:contains [ srs:userCount ?userTotalCount
         ; srs:topic ?topic] }
 4  FROM STREAM <http://sr.org/streams/S2> [RANGE 10m STEP 1m]
 5  FROM STREAM <http://sr.org/streams/S5> [RANGE 10m STEP 1m]
 6  FROM <http://sr.org/staticData/museum.rdf>
 7  WHERE {
 8     SELECT ?roomID ?userTopic (COUNT(?userID) AS ?userTotalCount)
 9     WHERE {
10        ?roomID srs:isConnectedTo ?roomID2 .
11        ?roomID2 srs:contains ?artworksID .
12        ?artworksID srs:topic ?artworkTopic .
13        ?userID srs:isInterestedIn ?userTopic .
14        ?artworkTopic skos:transitiveBroader ?userTopic.
15     }
16     GROUP BY ?roomID, ?topic
17     HAVING (?userTotalCount>10)
18     ORDER BY ?userTotalCount
19     LIMIT 5
20  }
```

The query produces as output the stream S6. A snapshot of the RDF triples flowing on S6 is presented in Listing 7.11.

Listing 7.11 An event in stream S6 produced by query Q3 and consumed by query Q4

```
<http://www.sr.org/data/room/31>
   a  srs:Room ;
   srs:contains [ srs:userCount "57"^^xsd:integer ;
      srs:topic topics:Renaissance_art ] .
```

In the case study, the guides' positions and the status (i.e., being free or occupied in a tour) are volunteered. Every time they enter a room of the museum, they can use a mobile app to let the system know of their status. Stream S7, presented in Listing 7.12, shows the information regarding the position and the status of every guide.

Listing 7.12 An event in stream S7 produced by the Guide Mobile App and consumed by query Q4

```
guides:54 srs:detectedAt  rooms:31 ;
          srs:status statuses:free .
```

Finally, the last query of the chain (Q4), presented in Listing 7.13, answers requirement R.4, i.e., recommending the free guides to organize a guided tour. Q4 is the most complex query in the case study; not only does it have multiple streams in input (Lines 4 and 5) that have to be processed together with static knowledge (Line 6), but it is also an *event-based* query. The DURING clause, at Line 9, allows to specify that the pattern preceding the DURING clause must be valid while the pattern following the DURING clause is valid. In this specific case, it checks that while there is a large enough group of people interested in a topic in a room, a guide able to talk about such a topic is present in the same room or in a room nearby.

Listing 7.13 Query Q4 that processes streams S6 and S7 and produces stream S8

```
1  REGISTER QUERY Q4 AS
2  PREFIX srs:<http://streamreasoning.org/ontologies/2014/7/srs#>
3  CONSTRUCT { ?roomID srs:contains [ srs:userCount ?userTotalCount
        ; srs:topic ?topic] }
4  FROM STREAM <http://sr.org/streams/S6> [RANGE 10m STEP 1m]
5  FROM STREAM <http://sr.org/streams/S7> [RANGE 10m STEP 1m]
6  FROM <http://sr.org/staticData/museum.rdf>
7  WHERE {
8     { ?roomId srs:contains [ srs:userCount ?userTotalCount ; srs:
          topic ?topic] . }
9     DURING
10    { ?guideID srs:status <http://sr.org/data/statuses/free> ;
11      srs:expertIn ?topic .
12      { ?guideID srs:detectedAt ?roomID . }
13        UNION
14      { ?guideID srs:detectedAt ?roomID2 . ?roomID2 srs:
            connectedTo ?roomID . }
15    }
16 }
```

Query Q4 pushes the recommendations for the guides on stream S8. An example of a recommendation is presented in Listing 7.14. The recommendations will appear on the mobile app of the guide who can decide to engage the visitors with a Blitz Guided Thematic Tour about Renaissance art.

Listing 7.14 An event in stream S8 produced by query Q4 containing the Blitz Guided Thematic Tour suggestions for the guides

```
guides:54 a srs:Guide ;
          srs:suggestedTour [ srs:userCount "54"^^xsd:integer ;
          srs:topic  topics:Renaissance_art ] .
```

7.5 Conclusions

The integration of heterogeneous data from different sources is gaining more and more attention, and when the data is characterized by a high degree of dynamism, as happens with data streams, new problems arise. This chapter discusses the semantic interoperability of data streams through RDF streams and RDF Stream Processing engines. It analyzes the data models, the query models, and the existing implementations. It highlights the features of such implementations and their points of strength. Finally, it presents a case study to show how to cope with data heterogeneity in a realistic scenario.

Research in RSP is still open, and new trends are emerging. Given the number of existing systems, a problem is their evaluation: there are several initiatives to compare the operators they support [38], their performances [27], and their ability to produce correct results [21]. The former highlights the differences in the query language syntaxes and semantics, and it was the starting point to an initiative to define shared standards in RSP, through a W3C Community Group.[20]

Reasoning on heterogeneous, incomplete, and noisy data streams is one of the most active topics in RDF Stream Processing. It investigates methods and techniques to define conceptual schemata for streaming data and to use them as drivers to access the data streams, to query them, and to integrate the continuous flow of answers. We saw that EP-SPARQL supports some Stream Reasoning features, while SPARQL$_{stream}$ is the first attempt for ontology-based data stream access. They are initial works and their support to Stream Reasoning techniques is still immature. IMaRS [20] and DynamiTE [37] are more advanced reasoning techniques to process RDF streams; they use ontologies to materialize the implicit data in data streams, coping with the dynamic and incomplete nature of the streams. Some works on Inductive Stream Reasoning also address the noisy nature of data streams [11], but more research is required.

References

1. Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.): The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, 21–25 October 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8219. Springer, Berlin (2013)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
3. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semantic Web **3**(4), 397–407 (2012)
4. Arasu, A., Babu, S., Widom, J.: The cql continuous query language: semantic foundations and query execution. VLDB J. **15**(2), 121–142 (2006)

[20] Cf. http://www.w3.org/community/rsp/

5. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Popa, L. (ed.) Proceedings of the Twenty-First ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Madison, 3–5 June 2002, pp. 1–16
6. Balduini, M., Della Valle, E., Dell'Aglio, D., Tsytsarau, M., Palpanas, T., Confalonieri, C.: Social listening of city scale events using the streaming linked data framework. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, 21–25 October 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8219, pp. 1–16. Springer, Berlin (2013)
7. Balduini, M., Bozzon, A., Della Valle, E., Huang, Y., Houben, G.J.: Recommending venues using continuous predictive social media analytics. IEEE Internet Comput. **18**(5), 28–35 (2014)
8. Barbieri, D.F., Della Valle, E.: A proposal for publishing data streams as linked data - a position paper. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M. (eds.) LDOW, CEUR Workshop Proceedings, vol. 628. CEUR-WS.org (2010)
9. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Proceedings of ESWC2010 (2010)
10. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Querying rdf streams with c-sparql. SIGMOD Rec. **39**(1), 20–26 (2010)
11. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Huang, Y., Tresp, V., Rettinger, A., Wermser, H.: Deductive and inductive stream reasoning for semantic social media analytics. IEEE Intell. Syst. **25**(6), 32–41 (2010)
12. Botan, I., Derakhshan, R., Dindar, N., Haas, L.M., Miller, R.J., Tatbul, N.: Secret: a model for analysis of the execution semantics of stream processing systems. Proc. VLDB **3**(1), 232–243 (2010)
13. Breslin, J.G., Decker, S., Harth, A., Bojars, U.: Sioc: an approach to connect web-based communities. Int. J. Web Based Communities **2**(2), 133–142 (2006)
14. Calbimonte, J.P., Corcho, Ó., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
15. Calbimonte, J.P., Jeung, H., Corcho, Ó., Aberer, K.: Enabling query technologies for the semantic sensor web. Int. J. Semant. Web Inf. Syst. **8**(1), 43–63 (2012)
16. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J. Autom. Reason. **39**(3), 385–429 (2007)
17. Compton, M., Barnaghi, P.M., Bermudez, L., Garcia-Castro, R., Corcho, Ó., Cox, S., Graybeal, J., Hauswirth, M., Henson, C.A., Herzog, A., Huang, V.A., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K.R., Passant, A., Sheth, A.P., Taylor, K.: The ssn ontology of the w3c semantic sensor network incubator group. J. Web Semant. **17**, 25–32 (2012)
18. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. W3C recommendation (2012)
19. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a streaming world! reasoning upon rapidly changing information. IEEE Intell. Syst. **24**(6), 83–89 (2009)
20. Dell'Aglio, D., Della Valle, E.: Incremental reasoning on RDF streams. In: Harth, A., Hose, K., Schenkel, R. (eds.) Linked Data Management. Chapman and Hall/CRC, Boca Raton (2014)
21. Dell'Aglio, D., Calbimonte, J.P., Balduini, M., Corcho, Ó., Della Valle, E.: On correctness in rdf stream processor benchmarking. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, 21–25 October 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8219, pp. 326–342. Springer, Berlin (2013)

22. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: rewriting and optimization. In: Abiteboul, S., Böhm, K., Koch, C., Tan, K.L. (eds.) International Conference on Data Engineering, pp. 2–13. IEEE Computer Society, Hannover (2011)
23. Gupta, A., Mumick, I.S. (eds.): Materialized Views: Techniques, Implementations, and Applications. MIT Press, Cambridge (1999)
24. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09), pp. 2040–2045. IJCAI, Pasadena (2009)
25. Klyne, G., Carroll, J.J.: Resource description framework (RDF): concepts and abstract syntax. W3C recommendation (2006)
26. Le Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (eds.) International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
27. Le Phuoc, D., Dao-Tran, M., Pham, M.D., Boncz, P.A., Eiter, T., Fink, M.: Linked stream data processing engines: facts and figures. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) International Semantic Web Conference (2). Lecture Notes in Computer Science, vol. 7650, pp. 300–312. Springer, Heidelberg (2012)
28. Lenzerini, M.: Data integration: a theoretical perspective. In: Popa, L. (ed.) Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Madison, 3–5 June 2002, pp. 233–246
29. McGuinness, D.L., Van Harmelen, F., et al.: OWL web ontology language overview. W3C Recommendation **10**(10), 2004 (2004)
30. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: OWL 2 web ontology language: structural specification and functional-style syntax. W3C Recommendation **27**(65), 159 (2009)
31. Ortiz, C.E.: An introduction to near-field communication and the contactless communication API. Oracle Sun Developer Network (2008)
32. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. ACM Trans. Database Syst. **34**(3), Article No. 16 (2009)
33. PrudHommeaux, E., Seaborne, A., et al.: SPARQL query language for RDF. W3C Recommendation (2008)
34. Rinne, M., Nuutila, E., Törmä, S.: Instans: high-performance event processing with standard rdf and sparql. In: Glimm, B., Huynh, D. (eds.) International Semantic Web Conference (Posters & Demos), CEUR Workshop Proceedings, vol. 914. CEUR-WS.org (2012)
35. Schreiber, F.: Is time a real time? An overview of time ontology in informatics. In: Halang, W., Stoyenko, A. (eds.) Real Time Computing. NATO ASI Series, vol. 127, pp. 283–307. Springer, Berlin/Heidelberg (1994). doi:10.1007/978-3-642-88049-0_14. http://dx.doi.org/10.1007/978-3-642-88049-0_14
36. Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond horn ontologies. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 1093–1098. IJCAI/AAAI, Barcelona (2011)
37. Urbani, J., Margara, A., Jacobs, C.J.H., van Harmelen, F., Bal, H.E.: Dynamite: parallel materialization of dynamic rdf data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 8218, pp. 657–672. Springer, Heidelberg (2013)
38. Zhang, Y., Pham, M.D., Corcho, Ó., Calbimonte, J.P.: Srbench: a streaming rdf/sparql benchmark. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 7649, pp. 641–657. Springer, Heidelberg (2012)

Part III
Social Networks as Information Sources

Pervasive information systems have to deal with a multitude of heterogeneous data that are mainly the result of two independent phenomena that reached critical mass at the same time: the advent of the *Internet of Things* and the increase in volume of user-generated content produced by *social networks* and smart mobile terminals. As an example, in a Cultural Heritage Pervasive Information System, the data may come from the most common social networks and express information on users' preferences or moods when they are observing a particular cultural item (e.g., a picture or a sculpture) or visiting a cultural site (e.g., museum or archaeological ruins).

Part III mainly focuses on *social data analysis*, a multidisciplinary research effort that aims at making sense of the activity of people and of the content they produce, use, and share within online social networks. Such problem is strictly related to crowdsourcing and constitutes the basis for research topics like opinion mining and sentiment analysis. On the other hand, security and privacy issues have to be considered when users' personal information is available on the Web.

Chapter 8 provides a preliminary description of the theoretical foundations behind a social network and the current state of the field. In addition, the authors present the GIVAS (global interactive virtual archaeological system) project, a multimedia environment for archaeologists, cultural heritage researchers, and normal tourists that provides a comprehensive collaborative platform for managing, searching, visualizing, and sharing multimedia cultural heritage information using an innovative social network approach.

The spread of social networks such as Twitter, Facebook, and Google+ or specialized ones such as LinkedIn and Viadeo allows sharing opinions on different aspects of everyday life that constitute a rich source of data for opinion mining. The interest that potential customers show in online opinions and reviews about products is something that vendors are gradually paying more and more attention to. In this scenario, a promising approach is sentiment analysis: the computational study of opinions, sentiments, and emotions expressed in a text. Its main aim is the identification of the agreement or disagreement statements that deal with positive or negative feelings in comments or reviews.

Chapter 9 addresses the opinion mining problem within social networks and discusses the related state of the art. A probabilistic approach based on the latent Dirichlet allocation (LDA) is then presented. The proposed method has been applied for the real-time analysis of documents, such as tweets or Facebook posts, in English and Italian language of opinion holders or social groups in the DATABENC context: urban spaces, museums, archaeological parks, and events.

As further aspect, the ever-increasing number of social network users, on the one hand, and the massive amount of information being shared daily, on the other, have encouraged attackers to develop and use different techniques to collect and analyze such information for a number of malicious purposes, including spear-phishing attacks and identity theft. Clearly, this trend represents a significant challenge for both users and administrators. In fact, the widespread adoption of online social networks has raised a wide range of security and privacy concerns, which have not been fully addressed yet. In many cases, users are not even aware of the disclosure of their personal information through their profiles. Leakage of a user's private information can happen in different ways, including inconsistent or faulty privacy setting and inference attacks.

Chapter 10 discusses the main security and privacy issues associated with online social networks and investigates some attack models used to reveal a user's private information. The authors also discuss different strategies and regulations that can prevent disclosure of private information through online social networks.

Chapter 8
Multimedia Social Networks for Cultural Heritage Applications: The GIVAS Project

Vincenzo Moscato, Antonio Picariello, and V.S. Subrahmanian

8.1 Introduction: Social Networks and Multimedia

Nowadays, it is well known that the tremendous growth in the use of online *social networks* over the past decade (from 8 % of all Internet users in 2005 to more than 72 % in 2014) suggests that these Internet communities are, and will continue to be, a dominant venue for social interactions [9]. In addition, the explosive use of these platforms marks the transition from text-only communications—the usual form of communication in primitive Internet-based social networks—to richer multimedia content exchanges.

This is also the era in which *multimedia* is progressively becoming content driven and goal oriented, thus promoting a large number of applications showing a deep user collaboration space.

The actual statistics are very impressive: in 2012, for example, every minute of the day, 100,000 tweets were posted; 48 h of videos were updated on YouTube with 2,800,000 views; 685,000 pieces of content were shared on Facebook; and 2,000,000 queries were made on Google.

While the amount of multimedia data on the Internet is growing continuously, it is extremely important for users not only to share multimedia content with each other but also to receive content they are interested in.

In this framework, a ubiquitous platform that makes it possible to link people and data in all forms (text, speech, audio, video, and so on) as in a social network

V. Moscato (✉) • A. Picariello
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, University of Naples Federico II, via Claudio 21, Napoli, Italy
e-mail: vmoscato@unina.it; picus@unina.it

V.S. Subrahmanian
UMIACS and CS, University of Maryland at College Park, UMD, College Park, MD, USA
e-mail: vs@umiacs.umd.edu

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_8

is highly desirable in a large number of real-life applications: a social network of media and persons, based on a robust infrastructure of camera sensors and mobile devices (e.g., smartphones, tablets), is quickly becoming a necessary platform for enabling user collaboration and information sharing [6, 7]. Nowadays, such an integration of multimedia services and social networks is starting to change users' lifestyles and way of working. Moreover, publicly accessible communities (i.e., Facebook, Twitter, and so on) have developed alongside limited-access social platforms such as Yammer and Socialcast, which provide corporate nets for business communication and carry these social interaction methods into the workspace.

From the point of view of the social sciences, the concept of the social network dates back to 1960, and the advent of the Internet era, and in particular of Web 2.0, provided a modern and efficient infrastructure for creating and managing interconnecting nets. Generally speaking, a social network is a structure of "nodes" (i.e., individuals, organizations, and so on) that are connected to each other using a special kind of "relation" (e.g., resources, values, point of interests, and friendship): social network theory is, in that regard, a multidisciplinary science that combines sociology, economics, and—of course—computer science and telecommunication sciences [15].

Distributed social networks are organized with flat architectures, that is, there is no super node providing global knowledge of the network. These kinds of social networks can be easily achieved using *device-to-device* (D2D) communication technology of next-generation mobile communication systems.

Notably, social multimedia networks are characterized by heterogeneous "entities" and "relations" [1, 16]. If we consider once again Facebook as an example, users, concepts, and multimedia objects coexist in the network with various types of relations: users may, for instance, share multiple relations such as, for example, friendship, group membership, or message communication. Multimedia objects may also have multiple relations, each of them reflects a similarity in certain features or attributes (e.g., color, shape, attributes, or metadata). In addition, each user can post comments or ratings for a multimedia object, bookmark it as a favorite, or even place notes in certain regions in the object, which form multiple relations between the user and the multimedia object [11, 14].

The richness of entities and relations provides a comprehensive description of the contextual information, facilitating the design of novel multimedia applications with new user experiences.

In a multimedia social network (MSN) community, a group of users forms a dynamically changing network infrastructure to share and exchange data and multimedia content, as well as other resources. For example, in a peer-to-peer file-sharing system, users pool together resources and cooperate with each other to provide an inexpensive, highly scalable, and robust platform for distributed data sharing.

However, since participation in many MSNs is often voluntary and unregulated, users' full cooperation cannot be guaranteed unless there exist powerful central authorities who mandate and enforce user cooperation.

In this chapter, we are interested in modeling and designing a special social network that connects "people" from the cultural heritage domain to "multimedia resources." In particular, we describe an MSN, i.e., a social network allowing users to store and share in a smart environment texts, audio, digital video, and pictures [16]. In addition, a cultural heritage social networker may be interested in the link between a real-life scenario and a virtual environment generated by powerful 3D engines.

Following the vision of [13], connecting people to the resources they need is a fundamental task for any society and for a large variety of applications: for example, connecting patients to the appropriate health-care facility, farmers to the right people in the right agricultural knowledge centers, and disaster victims to the best shelters is an action that every society and its agents (government, military, nonprofit organizations) carry out on a regular basis. In developing countries, needs for security, food, water, health care, education, and basic transportation remain unmet, while the proper technology could be used to help people find the "connections" needed to improve their lives.

In a real cultural heritage scenario, we can imagine people connected in a giant social network, where every node is a real-life entity (people, organizations, facilities, equipment, artifacts, monuments, and so on) and every edge is a local connection between node pairs that are spatially, emotionally, or logically close to them.

For an archaeologist, an MSN accessible from simple smartphones or computers constantly updates information about the physical entities being studied. For simple tourists, the virtual tour of an ancient ruin, together with all the comments of experts or other tourists, may enhance their cultural experience. In this way, the whole network acts as a "world observatory" and yet provides only information to people related to their circles.

This kind of network is, by definition, a dynamic entity. It is designed to be always current and constitutes a model of the evolving world; it captures events and situations in the life of each person or thing and collective events and situations that affect any segment of human society represented in the net.

In what follows, we will describe the *Global Interactive Virtual Archaeological System (GIVAS) project*, which aims at creating an MSN for cultural heritage applications. In particular, GIVAS provides a social virtual 3D environment that is strictly correlated to a real one by means of a set of sensors and allows different users to accomplish their tasks using social networking and multimedia facilities: an archaeologist can find help in classifying an artifact using multimedia descriptions of similar objects and comments/suggestions of other colleagues; a tourist can browse a virtual site displaying comments of other people and enjoy multimedia stories about objects of interest.

The chapter is organized as follows. Section 8.2 describes the GIVAS project, while Sect. 8.3 reports a case study in the cultural heritage domain. Finally, Sect. 8.4 draws some conclusions and outlines additional future work to extend the discussed project.

8.2 GIVAS Project

8.2.1 The System at a Glance

GIVAS, i.e., *global interactive virtual archaeological system*, is an MSN particularly suitable for archaeological sites that provide the following capabilities:

- It allows archaeologists from around the world to continuously "watch" the ongoing work on an ancient site from wherever they are in the world. This is done by having a number of cameras deployed on site streaming imagery back to GIVAS. Archaeologists are also allowed to change the "focus" of the cameras so that they can see what they want. Note that these cameras can also include a "public facing" version of GIVAS that allows potential tourists, amateur archaeologists, and virtually anyone else in the world to get a view of the site.
- It allows archaeologists to communicate with one another in real time, sharing opinions on what needs to be done, e.g., it allows to mark up part of the images of a particular location and to give inputs to on-site archaeologists.
- It allows the public to view not only the current state of the site but what it looked like at various stages of the excavation: in this way, these "virtual visitors" can make comments, communicate with the archaeologists, and become part of the MSN of archaeologists working on the site, virtually allowing them to be involved in the construction process at the site.

From a general perspective, we describe an MSN that manages data, comments, and "things" contributed by different categories of people, such as the following: (1) trusted archaeologists working on the excavation, (2) trusted archaeologists and others (e.g., collaborating computer scientists), (3) VIPs (like large donors, sponsors, ministers, high government officials, museum officials, etc.), and (4) members of the general public (e.g., tourists). This can be best done through the careful design of a social network that allows all these diverse groups to participate in the site, enriching it greatly.

Starting from these general views and going more in depth, in the GIVAS *world*, different kinds of users—in the 3D environment in the shape of *avatars*—can communicate and interact with other users and objects, in the 3D environment in the shape of *prims*, which are equipped with a *multimedia gallery* (e.g., video, audio, image, text, 3D models, etc.) and a *notice board*.

In other words, a GIVAS world allows to create a *Multimedia Social Graph* related to a physical site, in which *nodes* are basically composed of users, objects, multimedia data, and notice boards (correlated with an object and not with a user) from the cultural heritage domain and *edges* reflecting the particular relationships that can be established among nodes.

Figure 8.1 shows an example of a GIVAS social graph in which some possible relationships among the main entities are highlighted. As can be seen, a person can be a *friend* of another person or can belong to a given group. In addition, a user can create an object adding multimedia information (e.g., multimedia gallery) and

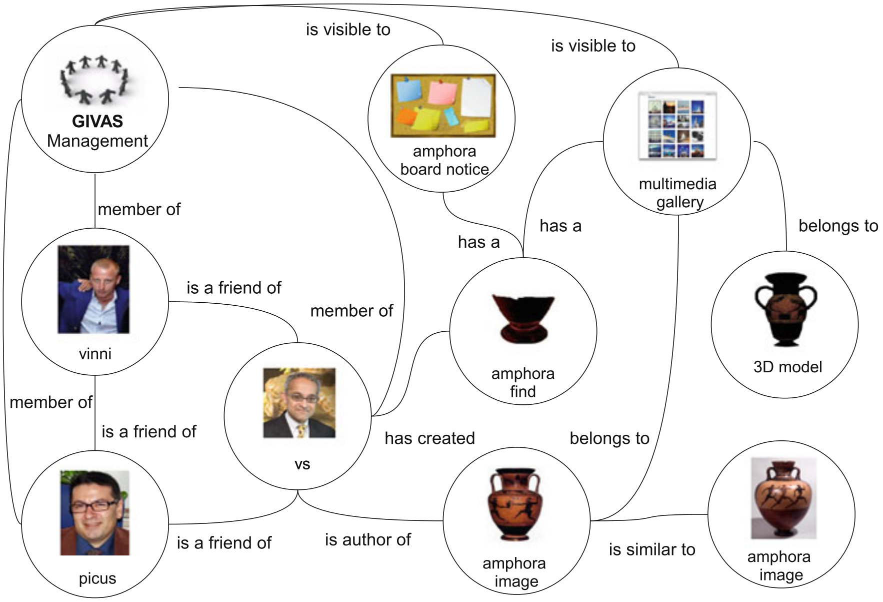


Fig. 8.1 GIVAS social graph

enabling a virtual notice board. Other users of the same group can, as an example, enrich the multimedia collection and post some comments on the objects.

8.2.2 GIVAS Architecture

The GIVAS architecture is inspired by the structure of a typical data pervasive system. In this case, data gathered by a set of sensors are exploited to provide a strict correlation between the real world (i.e., an archaeological site) and its virtual equivalent (i.e., a 3D reconstruction) using different rendering devices, from mobile to standard laptop and sophisticated cave systems. Additionally, the GIVAS system shows capabilities of a social and collaborative multimedia environment.

Figure 8.2 provides a functional overview of the system, while its main components are detailed in the following.

8.2.2.1 Site Manager

The *Site Manager* component with the related *sensors* (i.e., video cameras and radio-frequency identification (RFID) devices) allows a sort of abstraction of any kind of archaeological sites as a social networking environment, providing a

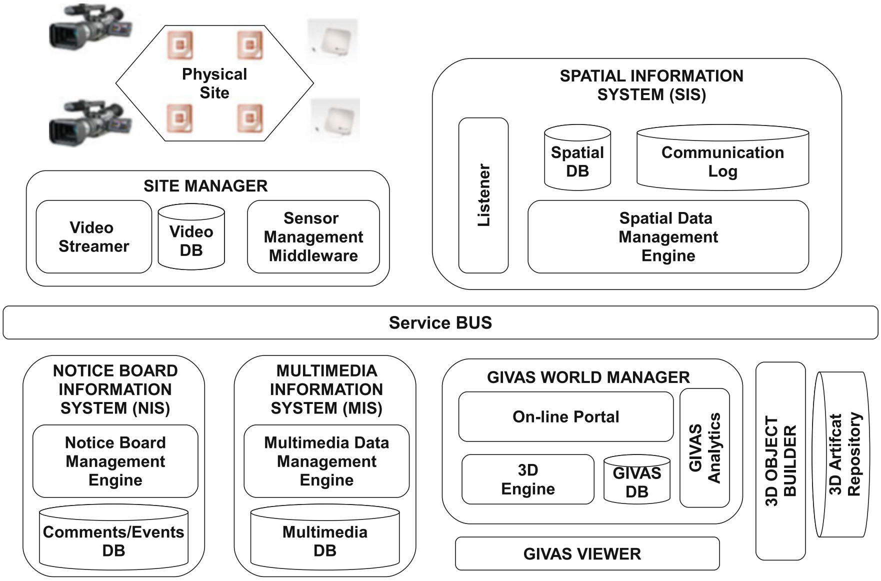


Fig. 8.2 GIVAS system architecture

set of basic functionalities to detect and monitor user behavior (e.g., actions of archaeologists or tourists) and objects (e.g., sculptures, finds, pictures) that are present within the physical site.

In particular, leveraging the RFID technology (i.e., tags and antenna), it is possible to supervise any possible change of objects' position at any time in order to guarantee in real time a "perfect" correspondence between the physical and virtual environments.

Furthermore, a set of video cameras is devoted to control several events such as the exit and entry of persons in different rooms/areas of the site or other kinds of actions. Moreover, an archaeologist can choose which kind of rooms, objects, and actions to monitor changing at any time the observation focus. To this purpose, proper *computer vision* facilities and *reasoning* techniques are opportunely exploited to recognize events of a simple semantics (i.e., "a person is in a room," "a person exits from a room," "a person is near an object," etc.).

The physical communication with sensors is managed by means of a *Sensor Management Middleware* module, which is able to gather heterogeneous information coming from the deployed devices and shows in the case of video cameras basic *event detection* capabilities.

A *video streamer* module (with the *Video DB*) is exploited to guarantee both the surveillance of rooms and remote view of the site.

8.2.2.2 Spatial Information System

The *spatial information system (SIS)* component allows to store and manage the entire cartography (in a 2D format) and specific locations of persons and objects related to a given archaeological site, providing a set of functionalities able to "link" at any time objects and persons to their effective spatial positions within the site.

In particular, the component is composed of a *listener* module that generates a proper instance for each sensor deployed within the environment. In particular, each listener instance manages a dedicated and virtual communication with a given sensor: RFID devices send to the related listener instance messages about the position of objects on which they are attached, while video cameras send a message with a particular event that has just been detected. The communication is carried out in a "triggered" way (i.e., messages are produced when an object changes its position or an event is recognized).

Once a message has been captured by a listener, it is forwarded with the sensor identifier to the *Spatial Data Management Engine* that updates locations of objects/persons within the *Spatial DB* and provides for communicating by the *Service BUS* a new event instance (related to an object or a person) with all the related information.

The Spatial Data Management Engine services are also invoked by the system *Service BUS* when a new object is created by the *Object Builder* or the *GIVAS Engine* and has to be located in a GIVAS *world*.

The system log is then stored in a *Communication Log* database that can be used to reconstruct change of positions in the course of time of all the objects and persons in the various stages of the excavation.

8.2.2.3 Notice Information System

The *notice information system (NIS)* component, on the one hand, reports using a set of virtual notice board events of interest for the monitored environment (e.g., a person is in a given room, an object has changed its position, a new object has just been created) and, on the other hand, allows to post and manage comments coming from users (e.g., archaeologists and tourists).

The component is basically composed of the *Notice Board Management Engine* module able to receive from the system *Service Bus* all events and comments that are then stored and indexed in the *Comment and Event* database.

The captured events and comments can be posted in virtual notice boards that are available in different rooms and on specific objects of a GIVAS world. In particular, the notice boards have two different sections: (1) *public*, in which comments are visible to all the users, and (2) *private*, in which comments are visible to specific users or groups of users.

Comments posted by users can be opportunely exploited to help an archaeologist in the classification and 3D reconstruction tasks of a find (e.g., a piece of an amphora).

At the same time, the module provides some services to quickly retrieve events and comments based on the different basic and advanced search filters.

8.2.2.4 Multimedia Information System

The *multimedia information system (MIS)* component allows managing all the different kinds of multimedia data (e.g., images, videos, audios, texts, 3D models, etc.) that can be attached to a particular object within the site.

It is composed of a *Multimedia Data Management Engine* that provides, on the one hand, a basic mechanism to index raw data in the *Multimedia DB* base on their features and, on the other hand, primitives of *content-based retrieval* based on both *low-level* and *high-level* characteristics.

For example, color, texture, and shape information features can be used to describe at a low level an image, while generic object metadata (i.e., keywords and tags) and other kinds of annotations can be exploited to characterize it from a high-level point of view. Computer vision algorithms able to determine the *similarity* between two images based on their low-level features can be used to implement *query by example* functionalities.

On the other hand, a different similarity notion based on the distance of metadata values in a given taxonomy can be adopted to realize a *semantic retrieval*.

8.2.2.5 GIVAS World Manager

The *GIVAS World Manager* component provides a set of functionalities to manage a 3D GIVAS *world* that constitutes an example of an MSN. The component is composed of several modules as described in the following.

The *Online Portal* allows users to create their accounts, define their profile, design their *avatar*, obtain general information about the GIVAS world, and view objects' notice boards/multimedia galleries and in real time different areas of the site using the deployed video cameras.

The *3D Engine* allows creating a specific GIVAS world by defining the terrain and the basic constituting objects (eventually provided with complex textures) and importing from external sources avatars with the related user profiles and prims models with the related multimedia annotations. In addition, the module has to manage *multiplayer* interactions among users—also providing basic chatting and message exchange mechanisms and interactions among users and objects (e.g., creating new objects, adding multimedia descriptions and virtual notice boards, posting comments, browsing of multimedia collections, and so on).

The *GIVAS DB* stores all the components of a GIVAS world: avatar with the related user information and prims and their annotations.

Services of SIS, NIS, and MIS components are invoked to determine the last locations of persons and objects, to retrieve multimedia data related to an object, and to access the comments in a virtual board notice.

A *GIVAS Analytics* module is eventually exploited to perform several statistics and analyses for management purposes.

8.2.2.6 Service BUS

The *Service BUS (SB)* component allows managing communication among the GIVAS components with respect to assigned protocols and data formats and in accordance to a message exchange model.

The goal is to realize secure communication among different software applications as in a *service-oriented architecture* (SOA): monitoring the routing of message exchange between services, resolving contention between communicating service components, controlling deployment, and versioning services.

8.2.2.7 GIVAS Viewer

The *GIVAS Viewer* component allows in an immersive way to browse a 3D virtual archaeological site.

In particular, as in the social network paradigm, it is possible to explore the environment by means of an avatar, add new friends, create groups, create new objects, interact with existing objects and browse the related multimedia collection, chat and interact with other users, post comments, receive notifications of events, and many other different activities.

8.2.2.8 Object Builder

The *Object Builder* component provides a set of functionalities that assists users in creating 3D objects with the related multimedia annotation and locating them in the virtual site. When it is necessary to create a new object, its services are invoked by the GIVAS 3D Engine.

In particular, the component provides a set of functionalities for retrieving similar objects from the *3D Artifact Repository* through a proper *Query Algebra* based on a *model graph approach* (reticula, faces, edges, and vertices) and also a set of facilities for storing additional information such as images, video, audio, or textual descriptions of the object.

8.2.3 Implementation Details

In this subsection, we report several implementation details for the GIVAS system components.

Concerning the GIVAS Site Manager component, the Sensor Management Middleware has been realized by extending and customizing the *SeNsIM* system that some of the authors have proposed in a previous work [10]. In particular, SeNsIM is an integration platform for heterogeneous sensor systems which makes the deployment of applications based on multiple sensor system/network possible and allows a generic user or an application to easily access data sensed by a network. SeNsIM provides a unique interface for local networks that allows a generic user to express queries by means of an intuitive query visual language. It was conceived with the aim of bridging the gap between heterogeneous sensor systems and of providing a generic user/application with a unique way of managing, querying, and interacting with them. Within the GIVAS project, SeNsIM was used for managing RFID and video camera sensor networks; in addition, reasoning techniques based on a stochastic graph representation of human activities were exploited to recognize events of interest [2–4, 12] for video data. Finally, the video streamer module has been implemented using facilities provided by *Apache Red5 Media Server*.

The SIS component has been realized by the *Quantum GIS* framework,[1] opportunely enriched with several "ad hoc" modules for managing interaction with the Site Manager and other GIVAS components.

The Notice Board Information System component has been entirely developed ex novo using *JAVA* technologies and exploiting *Cassandra* for storing comments and events. The MIS has been realized using facilities provided by the *Windsurf* system [8] that allows a multimedia content-based retrieval based on low- and high-level features for several kinds of data together with multidimensional indexing mechanisms. However, Service BUS functionalities are provided by the *WSO2*[2] Enterprise Service Bus.

Concerning GIVAS World Manager, the Online Portal module has been realized as web application by the *Drupal* content management system (CMS). The GIVAS 3D Engine is based on the *OpenSim*[3] Project; in turn GIVAS DB has been implemented combining functionalities provided by *MySQL* and *Postgres* database management systems (DBMSs). The analytics is then performed using some *Pentaho* modules.

The Object Builder facilities have been obtained by a *3D Building and Querying System* that some of the authors have proposed in a recent work [5]. The basic workflow supported by the system is the following: (1) a user enters a set of statements (this can also be generated in an automatic way using apposite Graphical User Interface (GUI) and Wizard facilities) describing the desired scene—she/he can also exploit already created objects (retrieving them by proper queries or searching them from web repositories and eventually associating a multimedia description and spatial coordinates); (2) the shell analyzes the statements and produces a tree parser which then is converted by the interpreter into a sequence of instructions using a set of predefined 3D operations; (3) the 3D object is then

[1] http://www.qgis.org/

[2] http://www.wso2.com/products/enterprise-service-bus/

[3] http://www.opensimulator.org/

created according to the mnemonic sequence entered; (4) the result is displayed on an output device; (5) the user can apply new transformations on the scene until satisfied. The objects can be finally stored with a pseudonym in the object database of the system.

Finally, we adopt the Second Life client as Givas Viewer

8.3 A Case Study

The GIVAS project originated through a collaboration by the University of Naples and the University of Maryland to support the work of archaeologists in classifying new finds coming from different archaeological sites using MSN and Virtual/Augmented Reality technologies.

In particular, we have chosen as GIVAS world the *San Marco* archaeological site, in Castellammare di Stabia near Naples (Italy), containing ruins of an ancient Roman villa where excavations are still active.[4]

As a preliminary step, we have performed a 3D reconstruction of the villa using a photogrammetry based approach using the GIVAS Engine facilities, we have generated the 3D environment related to the villa, shown in Fig. 8.3, and have mapped the objects' positions by the SIS component. Figure 8.4 shows a comparison between a fragment of the real site and the obtained virtual world.

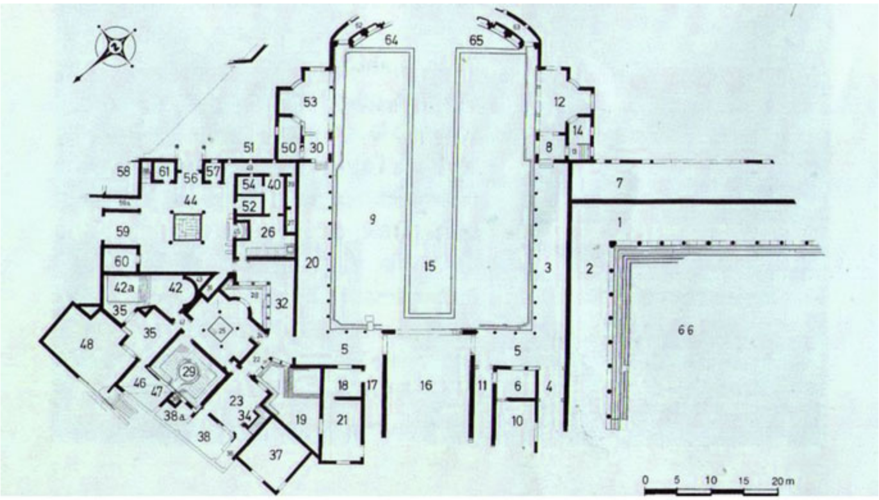


Fig. 8.3 Planimetry of the San Marco Villa

[4] We thank the support of Restoring Ancient Stabiae Foundation that provided access to the site.

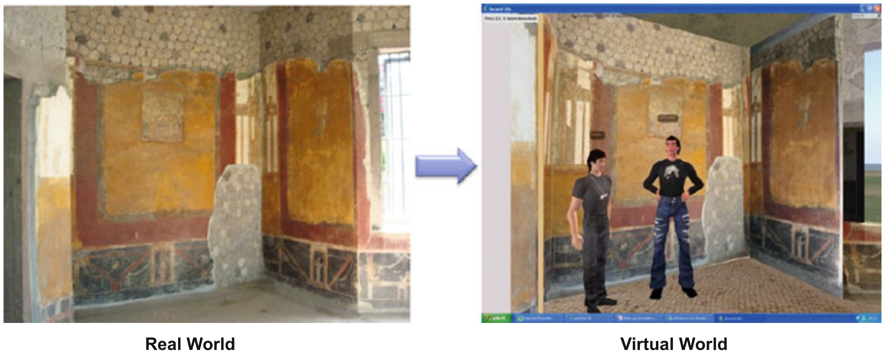


Fig. 8.4 Real vs. virtual

As a next step, we have attached to GIVAS the real set of RFID antennas and IP video cameras in order to monitor (1) the change of positions—within the same room or through different areas—of objects (each one is equipped with an RFID transponder) or (2) simple human behavior (i.e., a person who enters/exits in/from a room).

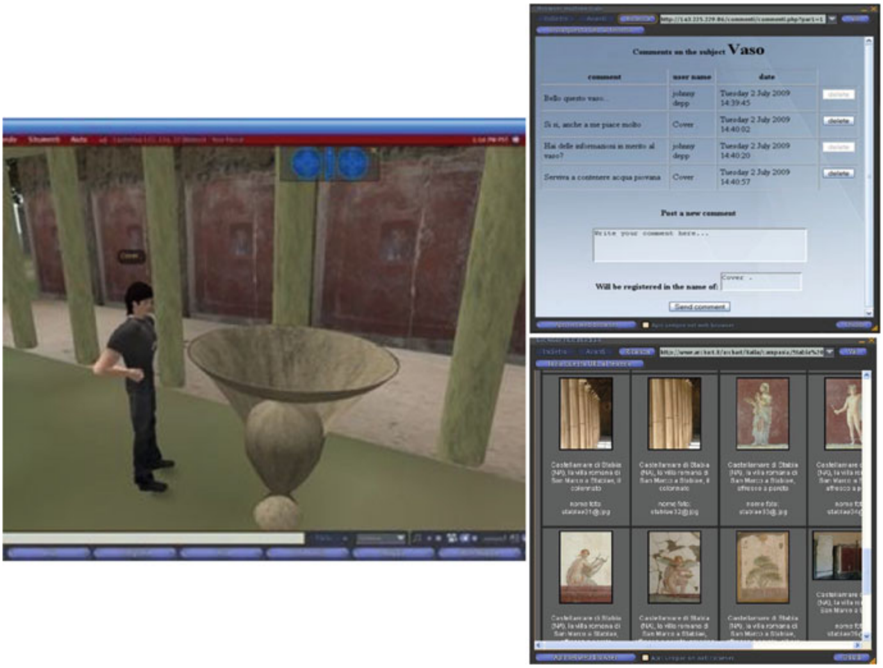
To this purpose, the Site Manager has been opportunely customized to integrate the two different sensor networks, to detect such kinds of events, and to allow a real-time site remote view from different perspectives.

Using the provided system functionalities, a real-world archaeologist who during excavation work discovers a new finding related to a specific object (e.g., a piece of an amphora) can:

- Create a 3D representation both of a finding and of the possible whole object containing the finding, using the Object Builder facilities via mobile device and 3D models available in the 3D Artifact Repository
- Add the 3D object to the GIVAS virtual world together with several multimedia information (e.g., some pictures just taken from the site, 3D models) and a notice board informing other archaeologists on the finding and asking them for possible help in the classification task
- Exploit MIS facilities to find other useful multimedia data (from other objects) to better characterize the finding
- Locate the object in a particular room together with other findings (e.g., warehouse) and add to it an RFID transponder to monitor possible changes of position
- Contribute to the physical reconstruction of the object to which—at the end of the process—it will correspond a new 3D object

On the other hand, other users from the virtual world can contribute to the classification work by:

- Analyzing the finding by browsing the related multimedia gallery
- Adding useful comments to the finding’s virtual notice board or other images to the finding’s multimedia gallery



Comments on the subject **Vaso**

comment	user name	date	
Bello questo vaso...	johnny depp	Tuesday 2 July 2009 14:39:45	delete
Si si, anche a me piace molto	Cover	Tuesday 2 July 2009 14:40:02	delete
Hai delle informazioni in merito al vaso?	johnny depp	Tuesday 2 July 2009 14:40:20	delete
Serviva a contenere acqua piovana	Cover	Tuesday 2 July 2009 14:40:57	delete

Post a new comment

Write your comment here...

Will be registered in the name of: Cover

Fig. 8.5 Notice board and multimedia gallery of an object

- Chatting online with other users and participating in virtual archaeological campaigns as in a multiplayer game to facilitate collaborative work

Figure 8.5 shows a user from the 3D virtual world who can access the notice board of an object or its multimedia gallery.

Eventually, the public can view not only the current state of the site but also what it looked like at various stages of the excavation or collaborate in the working activities.

8.4 Conclusions

In this chapter, we have described the main concepts related to social networks in the cultural heritage environment, in particular for archaeological sites. We have then described the GIVAS project, an MSN designed and implemented in a real environment. The actual implementation of GIVAS could benefit from several improvements. In particular, we think that GIVAS could (1) be embedded in a more general Smart City project, with more functionalities for tourists such as online ticketing, recommendation, and relations with other interesting touristic and cultural heritage sites; (2) benefit from more smart analytics such as the one proposed for

Big Data; (3) be provided with more sophisticated 3D environments, such as the ones offered by Augmented Reality using smart glasses particularly useful for on-site visits; and (4) be provided with more image processing functionalities in order to facilitate archaeologists' works.

These advancements could necessitate the release of a new version of GIVAS 2.0.

References

1. Aggarwal, C.C.: An Introduction to Social Network Data Analytics. Springer, Berlin (2011)
2. Albanese, M., Moscato, V., Picariello, A., Subrahmanian, V., Udrea, O.: Detecting stochastically scheduled activities in video. In: IJCAI, pp. 1802–1807 (2007)
3. Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V., Turaga, P., Udrea, O.: A constrained probabilistic petri net framework for human activity detection in video. IEEE Trans. Multimedia **10**(6), 982–996 (2008)
4. Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V., Turaga, P., Udrea, O.: A constrained probabilistic petri net framework for human activity detection in video. IEEE Trans. Multimedia **10**(8), 1429–1443 (2008)
5. Amato, F., Moscato, V., Picariello, A., Sansone, C.: Reconstruction of car crash scenes using a 3d building and query algebra. In: Proceeding of Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS), 2012, Sorrento, pp. 891–897, 25–29 November 2012. IEEE Computer Society, Silver Spring (2012) [ISBN 978-1-4673-5152-2]
6. Amato, F., Chianese, A., Moscato, V., Picariello, A., Sperli, G.: Snops: a smart environment for cultural heritage applications. In: Proceeding of the 12th International Workshop on Web 426 Information and Data Management (WIDM 2012), Maui, pp. 49–56, 02 November 2012. ACM, New York (2012) [ISBN 978-1-4503-1720-7]
7. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.: The role of social networks in information diffusion. In: Proceedings of the 21st International Conference on World Wide Web, 2012, Lyon, pp. 519–528. ACM, New York (2012)
8. Bartolini, I., Patella, M., Stromei, G.: The windsurf library for the efficient retrieval of multimedia hierarchical data. In: SIGMAP, pp. 139–148. Citeseer (2011)
9. Doreian, P., Stokman, F.: Evolution of Social Networks. Routledge, New York (2013)
10. Flammini, F., Gaglione, A., Mazzocca, N., Moscato, V., Pragliola, C.: Wireless sensor data fusion for critical infrastructure security. In: Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems, CISIS'08, Genova, 23–24 Oct 2008, pp. 92–99
11. Kazienko, P., Musial, K., Kajdanowicz, T.: Multidimensional social network in the social recommender system. IEEE Trans. Syst. Man Cybern. A Syst. Humans **41**(4), 746–759 (2011)
12. Molinaro, C., Moscato, V., Picariello, A., Pugliese, A., Rullo, A., Subrahmanian, V.: Padua: parallel architecture to detect unexplained activities. ACM Trans. Internet Technol. **14**(1), 3 (2014)
13. Narayanan, S., Li, A., Little, C.E., Gupta, N., Deng, P.X.: Integrated social network environment. US Patent 8,244,848, 2012
14. Rahman, M.A., Kim, H.N., El Saddik, A., Gueaieb, W.: A context-aware multimedia framework toward personal social network services. Multimedia Tools Appl. **71**(3), 1717–1747 (2014)
15. Scott, J.: Social Network Analysis. Sage, London (2012)
16. Zhang, Z., Wang, K.: A trust model for multimedia social networks. Soc. Netw. Anal. Min. **3**(4), 969–979 (2013)

Chapter 9
Sentiment Detection in Social Networks Using Semantic Analysis: A Tool for Sentiment Analysis and Its Application in Cultural Heritage Realm

Shi-Kuo Chang, Luca Greco, and Aniello De Santo

9.1 Introduction

Thanks to blogs, microblogs, social networks, and review collector sites, millions of messages appear daily on the Web. In general, this textual information can be divided into two main categories: facts and opinions. Facts are objective statements, while opinions reflect people's sentiments about products, other people, and events; the latter appear to be extremely important, as they are able to influence the decisional process [16]. The interest that potential customers show in opinions and reviews about products is something that vendors are gradually paying more and more attention to. Companies are interested in what customers say about their products as politicians are interested in how different news media are portraying them. In this scenario, a promising approach is the sentiment analysis: the computational study of opinions, sentiments, and emotions expressed in a text [11, 20]. In literature, there are many approaches to the sentiment analysis. A very broad overview of the existing work was presented in [14]. The authors describe in detail the main techniques and approaches for an opinion-oriented information retrieval. Early work in this area was focused on determining the semantic orientation of documents. In particular, some approaches attempt to learn a positive–negative classifier at

S.-K. Chang
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA
e-mail: chang@cs.pitt.edu

L. Greco (✉)
Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata,
Universita' degli Studi di Salerno, Via Giovanni Paolo II, 132 - 84084 Fisciano (SA), Italy
e-mail: lgreco@unisa.it

A. De Santo
DIETI, University of Naples Federico II, Naples, Italy
e-mail: aniello.desanto@gmail.com

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric
Systems and Applications, DOI 10.1007/978-3-319-20062-0_9

a document level. Turney [18] introduces the results of review classification by considering the algebraic sum of the orientation of terms as respective of the orientation of the documents. Starting from this approach, other techniques have been developed by focusing on specific tasks such as finding the sentiment of words [21]. Baroni [1] proposed to rank a large list of adjectives according to a subjectivity score by employing a small set of manually selected subjective adjectives and computing the mutual information of pairs of adjectives using frequency and co-occurrence frequency counts on the Web. The work of Turney [19] proposes an approach to measure the semantic orientation of a given word based on the strength of its association with a set of context-insensitive positive words minus the strength of its association with a set of negative words. By this approach, sentiment lexicons can be built and a sentiment polarity score can be assigned to each word [9, 12]. Sentiment polarity score means the strength or degree of sentiment in a defined sentence pattern. Artificial intelligence and probabilistic approaches have also been adopted for sentiment mining. In [15], three machine learning approaches (naïve Bayes, maximum entropy, and support vector machines (SVM)) have been adopted to label the polarity of a movie review dataset. A promising approach is presented in [11] where a novel methodology has been obtained by the combination of rule-based classification, supervised learning, and machine learning. In [17], an SVM-based technique has been introduced for classifying the sentiment in a collection of documents. Other approaches are inferring the sentiment orientation of social media content and estimate sentiment orientations of a collection of documents as a text classification problem [8]. More in general, sentiment-related information can be encoded by choosing specific words, as investigated in [13] where a lexicon for sentiment analysis has been obtained. In [22], a probabilistic approach called sentiment probabilistic latent semantic analysis (S-PLSA) is adopted for sentiment mining [10].

In this chapter, we discuss the adoption of an alternative approach to sentiment classification based on the latent Dirichlet allocation (LDA) [2]. In LDA, each document can be viewed as generated from a mixture of various topics: this is quite similar to probabilistic latent semantic analysis, except that in LDA the topic distribution is assumed to have a Dirichlet prior. By using the LDA approach on a set of documents belonging to a same knowledge domain, a mixed graph of terms (mGTs) can be automatically extracted [3, 4, 6]. Such a graph contains a set of weighted word pairs [7] that have proven to be discriminative for sentiment classification. In fact, LDA-based topic modeling is an effective conceptual clustering process, and it helps discover semantically rich concepts. Starting from these concepts that contain useful relationship indicators to identify the sentiment from messages, and by using a terminological Ontology Builder, which allows to identify the kind of semantic relationship between word pairs in mGT, the discussed system can accurately discover more latent relationships and make less errors in its predictions.

9.2 Extracting a Mixed Graph of Terms

We first explain how a mixed graph of terms (mGTs) can be automatically extracted from a document corpus to be used as a filter for sentiment extraction problems. An mGT is a complex structure allowing to capture and represent the information contained in a set of documents that belong to a well-defined knowledge domain; it can be defined as a graph $\mathbf{g} = \langle N, E \rangle$ where:

- $N = \{R, W\}$ is a finite set of nodes, covered by the set $R = \{r_1, \ldots, r_H\}$ whose elements are the *aggregate roots* (AR) and by the set $W = \{w_1, \ldots, w_M\}$ containing the *aggregates*. Aggregate roots can be defined as the words whose occurrence is most implied from the occurrence of all other words in the training corpus. Aggregates are defined as the words most related to aggregate roots from a probabilistic point of view.
- $E = \{E_{RR}, E_{RW}\}$ is a set of edges, covered by the set $E_{RR} = \{e_{r_1 r_2}, \ldots, e_{r_{H-1} r_H}\}$ whose elements are links between aggregate roots and by the set $E_{RW} = \{e_{r_1 w_1}, \ldots, e_{r_H w_M}\}$ whose elements are links between aggregate roots and aggregates.

  Two aggregate roots are linked if strongly correlated (in a probabilistic sense):

$$e_{r_i r_j} = \begin{cases} 1 \text{ if } \psi_{ij} \geq \tau \\ 0 \text{ otherwise} \end{cases}$$

  Aggregate roots can also be linked to aggregates if a relevant probabilistic correlation is present:

$$e_{r_i w_s} = \begin{cases} 1 \text{ if } \rho_{is} \geq \mu_i \\ 0 \text{ otherwise} \end{cases}$$

Details about mGT building and thresholds τ and μ_i will be further discussed. The feature extraction (FE) module is represented in Fig. 9.1. The input of the system is the set of documents:

$$\Omega_r = (\mathbf{d}_1, \cdots, \mathbf{d}_M)$$

After the preprocessing phase, which involves tokenization, stopword filtering, and stemming, a term-document matrix is built to feed the LDA [2] module. The LDA algorithm, assuming that each document is a mixture of a small number of latent topics and each word's creation is attributable to one of the document's topics, provides as output two matrices—Θ and Φ—which express probabilistic relations between topic document and word topic, respectively. Under particular assumptions [5], the LDA module's results can be used to determine the probability for each word v_i to occur in the corpus $W_A = \{P(v_i)\}$, the conditional probability between

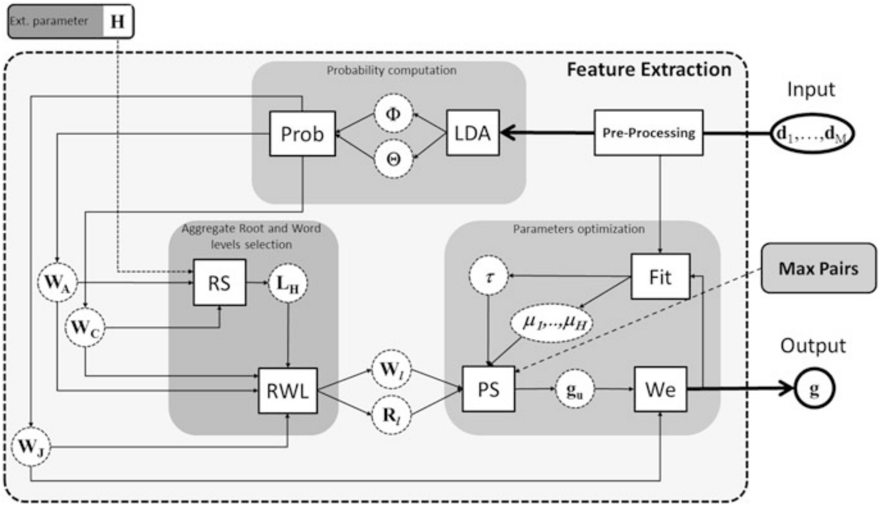


Fig. 9.1 A *mixed graph of terms* **g** structure is extracted from a corpus of training documents

word pairs $W_C = \{P(v_i|v_s)\}$, and the joint probability between word pairs $W_J = \{P(v_i, v_j)\}$. LDA and probability computation are discussed in detail in [2, 5, 6].

Defining *aggregate roots* (AR) as the words whose occurrence is most implied by the occurrence of other words of the corpus, a set of H aggregate roots $\mathbf{r} = (r_1, \ldots, r_H)$ can be determined from W_C :

$$r_i = \text{argmax}_{v_i} \prod_{j \neq i} P(v_i|v_j) \tag{9.1}$$

This phase is referred to as root selection (RS) in Fig. 9.1. A weight ψ_{ij} can be defined as a degree of probabilistic correlation between AR pairs: $\psi_{ij} = P(r_i, r_j)$. We define an *aggregate* as a word v_s having a high probabilistic dependency with an aggregate root r_i . Such a dependency can be expressed through the probabilistic weight $\rho_{is} = P(r_i|v_s)$. Therefore, for each aggregate root, a set of aggregates can be selected according to higher ρ_{is} values. As a result of the root-word level (RWL) selection, an initial mixed graph of terms structure, composed of H aggregate roots (R_l) linked to all possible aggregates (W_l), is obtained. An optimization phase allows to neglect weakly related pairs according to a fitness function discussed in [5]. Our algorithm, given the number of aggregate roots H and the desired max number of pairs as constraints, chooses the best parameter settings τ and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_H)$ defined as follows:

1. τ: the threshold that establishes the number of *aggregate root/aggregate root* pairs. A relationship between the aggregate root v_i and aggregate root r_j is relevant if $\psi_{ij} \geq \tau$.

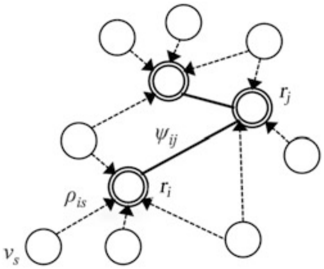


Fig. 9.2 Graphical representation of a *mixed graph of terms* structure

2. μ_i: the threshold that establishes, for each aggregate root i, the number of *aggregate root/word* pairs. A relationship between the word v_s and the aggregate root r_i is relevant if $\rho_{is} \geq \mu_i$.

   An mGT graphical representation is shown in Fig. 9.2.

9.3 Searching the Sentiment by the Use of the Mixed Graph of Terms

The discussed method adopts the mixed graph of terms for building a sentiment detector able to label a document according to its sentiment. The architecture is composed of the following modules:

- *Mixed graph of terms building module*: This module builds a mixed graph of terms starting from a set of documents belonging to a well-defined knowledge domain and previously labeled according to the sentiment expressed in them. In this way, the obtained mixed graph of terms contains information about the words and their co-occurrences representing a certain sentiment in a well-defined knowledge domain. Thanks to the LDA approach, such a graph can be obtained by the use of a set of few documents. The output of this module is a mixed graph of terms representing the documents and their sentiment. By feeding this module with positive or negative training sets, it will be possible to build mixed graphs of terms for documents that express positive or negative sentiment in a well-defined domain.
- *Sentiment mining module*: This module extracts the sentiment from a document thanks to the use of the mixed graph of terms as a sentiment filter. The input of this module is a generic document; the mixed graph of terms representing positive and negative sentiment in a knowledge domain and the output is the sentiment detected in the input document.

The sentiment extraction is obtained by a comparison between document and the mixed graph of terms according to the following algorithm:

- *Input of the algorithm*:
  - A set of comments, reviews about items, or social posts
  - The sentiment-oriented mixed graphs of terms mGT$^+$ and mGT$^-$ obtained by analyzing the (positively and negatively) training comments
  - An annotated lexicon L
- *Output of the algorithm*:
  - The average probabilities P^+ and P^- which express the probability that a sentiment, extracted from the set of comments or posts, is "positive" or "negative"
- *Description of the main steps*:
  1. For each word in the mGT$^+$ and the mGT$^-$, their synonyms are retrieved through the annotated lexicon L. In this case, *WordNet*[1] has been selected as lexicon.
  2. For each comment f_i, the probabilities $P^+_{f_i}$ and $P^-_{f_i}$ are determined as:

     $$P_{f_i}^{+/-} = \frac{(A + B + C + D)}{4}$$

     where A is the ratio between the sum of occurrences in the comment of words that are Aggregate Root Nodes and the total number of the Aggregate Root Nodes in the (positive/negative) mGT, B the ratio between the sum of occurrences in the document of words that are Aggregate Nodes and the total number of the Aggregate Nodes in the (positive/negative) mGT, C the ratio between the sum of the co-occurrence probabilities of Aggregate Root Nodes pairs that are in the document and the sum of all the co-occurrence probabilities of Aggregate Root Nodes pairs in the (positive/negative) mGT, and D the ratio between the sum of the co-occurrence probabilities of Aggregate Nodes pairs that are in the document and the sum of all the co-occurrence probabilities of Aggregate Nodes pairs that are in the (positive/negative) mGT.
  3. For each item, the probabilities P^+ and P^- are determined as:

     $$P^+ = \sum_i \frac{P^+_{f_i}}{\text{num\_of\_comments}}$$

     $$P^- = \sum_i \frac{P^-_{f_i}}{\text{num\_of\_comments}}$$

[1] http://wordnetweb.princeton.edu/perl/webwn

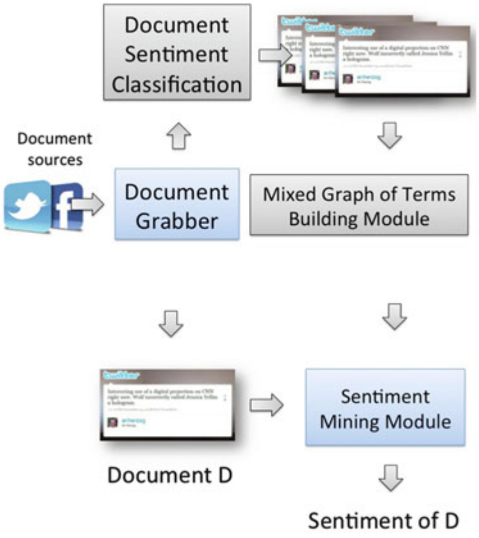


Fig. 9.3 System architecture for synchronous classification

The discussed approach is effective in an asynchronous sentiment classification but can also work in a synchronous way. In Fig. 9.3, the synchronous sentiment real-time classification architecture is depicted. For real-time working, two new modules are introduced:

- *Document grabber*: This module aims to collect documents from Web sources (social networks, blogs, and so on). These documents can be collected both for updating the training set and for their classification according to the sentiment. The training set update is an important feature of the proposed approach. In this way, in fact, the various mGTs can be continuously updated and improve their discriminating power introducing new words and relations and deleting inconsistent ones.
- *Document sentiment classification*: The new documents inserted into the training set have to be classified by the support of an expert. The aim of this module is to provide a user-friendly environment for the classification, according to their sentiment, of the retrieved documents.

9.4 A Case Study for Cultural Heritage Applications

We tested the effectiveness of the mGT-based approach in a cultural heritage context. A dataset consisting of 1000 posts has been collected from TripAdvisor: it contains 500 positive comments and 500 negative comments about monuments,

Table 9.1 Details about mixed graphs of terms extraction

	Sentiment	#_Concepts	#_Comments
MGT_p1	Positive	4	50
MGT_p2	Positive	5	100
MGT_p3	Positive	6	150
MGT_n1	Negative	4	50
MGT_n2	Negative	5	100
MGT_n3	Negative	6	150

Table 9.2 Details about the obtained mixed graphs of terms

	#_Comments	Accuracy (%)
Dataset_p1	450	78.23
Dataset_p2	400	83.52
Dataset_p3	350	86.91
Dataset_n1	450	79.43
Dataset_n2	400	84.27
Dataset_n3	350	85.67

landscapes, books, works of art, and artifacts. For each post, the following information are provided:

- *Review*: the comment written by the user
- *Rating*: a rating suggested by the user in the range 1–5
- *Sentiment*: the sentiment related to the comment; five experts of computational linguistics manually labeled the reviews

At this point, six mGTs have been built by selecting random comments from the dataset: three of them related to positive comments and the other ones related to negative comments. Table 9.1 provides more details about the mixed graph of terms extraction.

The sentiment classification on the remaining comments (test set) is conducted by the use of these mixed graphs of terms. Six datasets are considered, and the obtained results are depicted in Table 9.2.

In order to better evaluate the behavior of the system, a performance analysis when varying the length of the reviews is performed. In particular, the datasets are divided into four classes: comments with length included in the range 0–300 characters, comments with length included in the range 301–500 characters, comments with length included in the range 500–1000 characters, and comments with length over 1000 characters. The results obtained are depicted in Fig. 9.4.

In general, this experimentation shows that the system has better performance when the comments are not too long; in this case, the discriminating power of the mGT can be better observed. When the comment is longer, the great number of retrieved words can weaken the ability of the mGT in recognizing the sentiment.

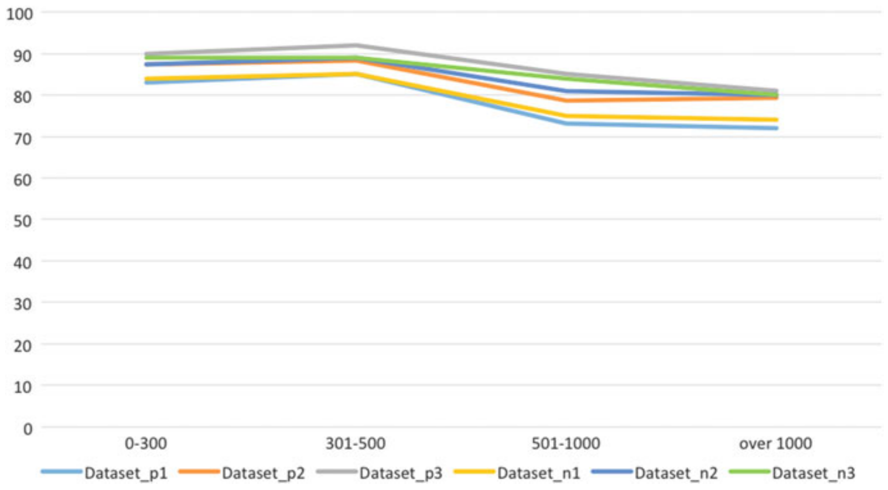


Fig. 9.4 Accuracy trend when varying the length of the reviews

Another reason for this worsening is that users often give a sharper opinion in a short comment. From the experimental campaign, some considerations can be made:

- The system performance improves with the increase of the root nodes. The reason is quite clear: the root nodes are able to better characterize the knowledge domain and the mood of the various comments.
- The system performance improves with the increase of the training set size. In this case too, the reason is clear: more comments provide more words and enhance the discriminating power of the mixed graph of terms.
- The system works well when the reviews have a number of characters under 500. In this case, the mGT is more effective in the retrieval of the mood thanks to the connections among the words. When the number of the words increases, the system cannot discriminate among the connections and their mood.

9.5 Conclusions

In this chapter, we investigated the use of the mixed graph of terms (mGTs) structure for the sentiment classification of textual documents. Once reference mGTs have been learned from training documents having a given sentiment orientation, the classification of a new document can be performed by using such reference mGTs as sentiment filters. Such a method has the strength to be language independent, relying mainly on a probabilistic technique. We evaluated the effectiveness of this approach by using a dataset related to the cultural heritage domain and collected from TripAdvisor. The results are encouraging and show that the method can contribute in an interesting way to the field of cultural heritage pervasive systems.

References

1. Baroni, M., Vegnaduzzo, S.: Identifying subjective adjectives through web-based mutual information. In: Proceedings of the 7th Konferenz zur Verarbeitung Natrlicher Sprache (German Conference on Natural Language Processing) KONVENS04, pp. 613–619 (2004)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Clarizia, F., Colace, F., Greco, L., Santo, M.D., Napoletano, P.: Improving text retrieval accuracy using a graph of terms. In: DMS, pp. 42–47. Knowledge Systems Institute (2011)
4. Clarizia, F., Colace, F., De Santo, M., Greco, L., Napoletano, P.: Mixed graph of terms for query expansion. In: 2011 11th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 581–586 (2011)
5. Colace, F., De Santo, M., Greco, L.: Weighted word pairs for text retrieval. In: Proceedings of the 3nd Italian Information Retrieval (IIR) CEUR Workshop Proceedings (2013)
6. Colace, F., De Santo, M., Greco, L., Napoletano, P.: Text classification using a few labeled examples. Comput. Hum. Behav. **30**, 689–697 (2014)
7. Colace, F., De Santo, M., Greco, L., Napoletano, P.: Weighted word pairs for query expansion. Inf. Process. Manag. **51**(1), 179–193 (2015)
8. Colbaugh, R., Glass, K.: Estimating sentiment orientation in social media for intelligence monitoring and analysis. In: 2010 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 135–137 (2010)
9. Gamon, M., Aue, A.: Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In: Arbor, A. (ed.) Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing, pp. 57–64. Michigan Association for Computational Linguistics, Ann Arbor (2005)
10. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of Uncertainty in Artificial Intelligence, UAI99, pp. 289–296 (1999)
11. Liu, B.: Sentiment analysis and subjectivity. In: Handbook of Natural Language Processing, 2nd edn. Taylor and Francis Group, Boca Raton (2010)
12. Neviarouskaya, A., Prendinger, H., Ishizuka, M.: Sentiful: a lexicon for sentiment analysis. IEEE Trans. Affect. Comput. **2**(1), 22–36 (2011)
13. Neviarouskaya, A., Prendinger, H., Ishizuka, M.: Sentiful: a lexicon for sentiment analysis. IEEE Trans. Affect. Comput. **2**(1), 22–36 (2011)
14. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
15. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up sentiment classification using machine learning techniques. In: Proceedings of Empirical Methods in Natural Language Processing, EMNLP, pp. 79–86 (2002)
16. Sebastiani, F.: Machine learning in text categorization. ACM Comput. Surv. **34**, 1–47 (2002)
17. Shein, K.P.P.: Ontology based combined approach for sentiment classification. In: Proceedings of the 3rd International Conference on Communications and Information Technology, CIT'09, pp. 112–115. World Scientific and Engineering Academy and Society, Stevens Point (2009)
18. Turney, P.D.: Thumbs up or thumbs down: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, Stroudsburg, pp. 417–424 (2002)
19. Turney, P., Littman, M.: Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Technical report, nrc technical report erb-1094, Institute for Information Technology, National Research Council Canada (2002)
20. Wang, C., Xiao, Z., Liu, Y., Xu, Y., Zhou, A., Zhang, K.: Sentiview: sentiment analysis and visualization for internet popular topics. IEEE Trans. Hum. Mach. Syst. **43**(6), 620–630 (2013)

21. Wilson, T., Wiebe, J., Hwa, R.: Just how mad are you? Finding strong and weak opinion clauses. In: Proceedings of the 19th National Conference on Artificial Intelligence, AAAI'04, San Jose, from 25 to 29 July 2004, pp. 761–767. AAAI Press, San Jose (2004)
22. Yu, X., Liu, Y., Huang, X., An., A.: Mining online reviews for predicting sales performance: a case study in the movie domain. IEEE Trans. Knowl. Data Eng. **24**(4), 720–734 (2012)

Chapter 10
Security and Privacy Issues in Social Networks

Sepideh Deliri and Massimiliano Albanese

10.1 Introduction

In recent years, *online social networks* (OSNs) like *Facebook*, *LinkedIn*, and *Twitter* have turned out to be fundamental parts of our online lives, and their popularity is increasing at a surprising rate each and every day. According to a survey conducted by the *Pew Research Center's Internet & American Life Project*, almost 72 % of online US adults are members of online social network sites, compared to only 8 % in 2005 and 67 % in late 2012 [3]. Among the social networking sites, *Facebook*—with nearly 1.28 billion users worldwide as of the first quarter of 2014—has been one of the most successful ones in recent years [14]. Users employ OSNs as a tool to connect with family, friends, colleagues, associates, or people with the same interests for different purposes such as professional networking, advertising their brands and businesses, job searches, making profit, or entertainment.

Besides the revolution that OSNs have generated in social networking, they have introduced new threats to their users due to their attractiveness, the ever-increasing number of users, and the massive amount of personal information they share. Being a part of users' daily lives, online social networks introduce new security concerns especially because of the potential exposure of huge amounts of personal information. Some examples of threats to OSNs include stealing the identities or credentials of users and social networking risks like *phishing*, *social engineering* and *reverse social engineering*, *spoofing*, *fiscal fraud*, *malware*, and *spam*. As a result, maintaining security and privacy of OSN users has become a major challenge in recent years.

S. Deliri • M. Albanese (✉)
George Mason University, 4400 University Drive, Fairfax, VA 22030, USA
e-mail: sdeliri@gmu.edu; malbanes@gmu.edu

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_10

In this chapter, we discuss the main classes of security and privacy attacks to online social networks and the countermeasures that can be used to protect the privacy of OSN users and keep shared data secure against different types of attacks. Additionally, we propose a set of guidelines for OSN users to protect themselves against different types of attacks. Finally, we examine existing laws and regulations introduced to combat online social network attacks and prosecute illegal activities by cyber criminals.

10.2 Overview of Online Social Networks

Online social networks have seen a rapid increase in the number of users, especially in the last few years [8]. Being a strong communication channel, online social media has become a popular tool to comment on worldwide events, improve information discovery, and facilitate social interaction. In addition, OSNs are powerful tools to not only stay connected with friends and family members but also to connect with professional groups. Additionally, they can be used for marketing and advertising purposes and for enhancing business reputation.

People join OSNs, create their own profiles, and connect to other members of the network with the main purpose of keeping in touch with them and sharing common interests. Most online social network sites share some common features. One of the fundamental properties of OSNs is that they allow their users to create their own profiles. Using this feature, OSN members can represent themselves with the purpose of connecting to other users in the network. Users can post contents like pictures, videos, and personal blogs in their profiles, which can be seen by all or only selected members of the network. Another common feature of OSNs is the ability to create links with other members of the network. If a link exists between two members, they are supposed to be neighbors. Neighboring users usually have common interests, and a higher degree of trust exists among them. Users can trace a specific OSN member by following the links connecting OSN users in order to reach his online profile. Finally, unlike other online sites that are usually content based, OSNs are mainly organized around their users. In other words, two major components of OSNs are user accounts and the links between users [12].

Although a social networking site can be a good way of connecting to people with similar interests, the ever-increasing number of OSN users leads to a growing amount of available data related to social relations. The availability of massive amount of personal information has attracted the attention of malicious users and motivated them to initiate attacks against OSNs by gaining access to the information shared by the users. Therefore, the privacy of OSN users can be at risk. Indeed, OSNs can become a tool that opens up new ways for criminals and hackers to perform undesired or fraudulent activities such as *spamming*, attacking through *viruses*, *phishing*, etc., which often results in *information and identity theft*.

10.3 Security and Privacy Threats

Online social media can introduce new threats for their users because of the potential for accessing a vast amount of personal information disclosed by OSN users themselves. Different types of assets are prone to attacks in OSNs, including private information of the individuals or organizations, digital identity, financial assets, intellectual property (IP), and corporate secrets and resources [7].

OSNs can be the target of several types of attacks like *phishing*, *spamming*, *clickjacking*, and *cross-site scripting*, to name a few. In this section, we discuss some common attacks on online social networks.

10.3.1 Social Engineering and Reverse Social Engineering Attacks

The main driver behind social engineering attacks is the fact that OSN users are not aware of the real value and importance of private information and, as a result, they do not pay much attention to keep it safe against attacks launched by malicious users. In such a situation, the attacker can deceive users into revealing confidential information employing different mechanisms.

A reverse social engineering attack is another threat to OSNs in which the malicious attacker deceives the user into contacting him using different types of techniques. Since the user initiates the connection, a higher degree of trust is then established between the attacker and the user. After this connection is established with the deceived user, the attacker begins his malicious actions, such as *phishing* and *spamming* [10].

In March 2012, a *Facebook social engineering* attack occurred in which the attackers created a fake Facebook account under the name of Admiral James Stavridis, NATO's Supreme Allied Commander Europe (*SACEUR*), in order to intrigue his friends and colleagues to contact him and deceive them to disclose confidential information using *social engineering* techniques. After the profile was reported to Facebook, it was taken down and an investigation began to identify the perpetrators.

Social engineering attacks can be accomplished using different mechanisms. For instance, *phishing* is a technique used to entice users to do things that are desirable for the attackers. The malicious user creates fake emails or websites to impersonate a trustworthy entity like a bank in order to deceive users to reveal private information. *Pretexting* is another technique used to perform a *social engineering* attack. With this technique, the attacker entices users to disclose confidential information or carry out actions that are desirable for the attacker by using a lie or false motive, known as *pretext*.

10.3.2 Identity Theft

Identity theft is a type of attack on OSNs in which the adversary attempts to collect personal information of OSN users so that he can impersonate the victim of the attack in order to gain some benefits or harm the victim. Different methods can be used to launch *identity theft* attacks, including *phishing*, accepting friend requests from unknown people, sharing account details with others, clicking links that lead the user to other websites, downloading free applications, low privacy settings, etc. This type of attack to OSNs can originate from both inside and outside the network. An insider attack is launched by a legitimate user of the social network who begins to behave in a malicious way. On the other hand, other types of attacks are launched by malicious users outside of the OSN.

Phishing is probably the most common method to initiate an identity theft attack. In this type of attack, the fraudulent user attempts to steal the victim's personal information—like his credentials or credit card information—by impersonating a legitimate user or entity such as a colleague or a bank.

Spear phishing is a special type of *phishing* attack in which the attacker targets a limited group of high-profile individuals, expecting to extract special information about such users. The key factor for the success of a *spear-phishing* attack is familiarity with the target users. Having some knowledge about the user, such as his email address, the attacker sends out an email to the victim, which seems to be sent from a legitimate user, with the intention of stealing personal information about the target of the attack like his password, credit card information, etc.

As an example, we can mention the attack on RSA, the security team of EMC, in March 2011. They reported information disclosure in mid-March due to a *spear-phishing* attack executed by exploiting an unpatched Adobe Flash vulnerability [1]. To launch the attack, the hackers targeted a small group of RSA employees and sent them two *spear-phishing* emails. It was sufficient that one of the employees opened the email containing the infected file. The *Trojan* embedded in the email exfiltrated information from the compromised account, including credentials that were later used by the attackers to access the targeted systems without being detected.

In order to increase the chance of success in a *spear-phishing* attack, the attacker may perform *reconnaissance* activities against the target. In other words, the attacker begins to collect information about the targets using different sources like *Open-Source Intelligence* (OSINT) and online social networks.

The attacker also needs the users' email addresses to launch *spear-phishing* attacks. He can search the victims' online profiles or attempt to guess the email address by using common email formats. According to an investigation conducted by *Trend Micro Research* in 2012, almost half of the email addresses of *spear-phishing* targets can be easily found by performing a simple Google search, and most of the others can be discovered by combining the user's name with his company's email domain [15].

10.3.3 Spamming Attacks

The main purpose of *spammers* is to send out a huge number of emails to advertise and sell their products and gain sensitive information about the users (e.g., username and password) by masquerading as a trusted party (e.g., banks or online payment processors). Such sensitive information can be gained using fake OSN profiles created by malicious users. With this approach, the attacker sends random friend requests to the members of the OSN target community waiting for them to accept their requests.

A *3-clique attack* can be used as a technique to find out the most vulnerable OSN members. After befriending any of the weak members of the community, the attacker sends friend requests to the weak member's friends [13]. Accepting the attacker's friend requests by any of the weak members' friends also increases the chance of the attacker to compromise the privacy of less accessible members of the OSN. Using this technique, information shared by the OSN members becomes vulnerable to *infiltration* attacks. This type of attack is a powerful tool used by attackers to carry out *spamming* and *phishing* attacks [17].

10.3.4 Malware Issues

Malware is a type of malicious software designed to cause damage to computer systems or take control of their operation in order to obtain sensitive information. Malware can easily spread in OSNs because of the main property of online social networks, which is connecting a huge number of users. Being a popular communication infrastructure, online social networks create a platform for the malware to easily propagate among their users and compromise more members.

For instance, *Koobface* was the first malware to propagate in many OSNs, including Facebook and Twitter. In 2008, Koobface originally spread on Facebook by delivering messages from an infected user to his Facebook friends. The message could also be sent from a fake account created by Koobface itself, which befriends OSN users automatically. A URL was embedded in the message that guided the intrigued users into a spoofed Facebook page and tricked them to install the Koobface malware. The main purpose of Koobface was to steal personal information of the users, including their log-in information, and launch some malicious activities, such as sending spam messages [7].

10.3.5 Clickjacking, Likejacking, and Cursorjacking Attacks

Clickjacking, also known as *UI-Redress* attack, is an example of *confused deputy* problem. In this type of attack, a malicious user deceives OSN users into clicking on a link, which is different from what the users expect it to be by overlaying multiple frames and hiding some frames from them. In other words, the webpage that is displayed to the user is different from the page where the user's action is taking place. Actually, the attacker chooses the webpage and clicking on a button on the page performs a different function than the user perceives.

Cursorjacking is another type of clickjacking attack, which deceives users into clicking on a malicious link and performing unwanted actions by changing the location of the cursor from the place that the user perceives to the location that the attacker expects.

In *likejacking*, the attacker uses *social engineering* techniques in order to intrigue users to click on links that include embedded scripts or code. The embedded links will then repost the attacker's link on the user's wall automatically or make the user like a survey page from which the attacker makes profit without the permission of the affected user.

As an example, we can mention the Facebook clickjacking attack that occurred in December 2009. In this attack, which spread through the http://fb.59.to website, a Facebook user was intrigued to click on a link posted on a friend's wall with the thumbnail and the caption "New Pix." Clicking on the link guided the user to a YouTube page while the link was also posted to the Facebook user's wall.[1]

10.3.6 Cross-Site Scripting

Cross-site scripting is a type of attack against Web applications, including OSNs, in which the attacker injects malicious code into target webpages and encourages the user to run the code in order to steal his sensitive information.

Cross-site scripting has become more common since HTML and AJAX were integrated. This is mainly due to the fact that the attacker does not need to deceive the user into clicking on a malicious link to exploit XSS vulnerabilities. In other words, a browser can send out HTTP requests on behalf of the user by using AJAX technology [6].

Attackers can use XSS scripting in online social networks to create *XSS worms*, also known as *XSS viruses*. This malicious code can spread among the visitors of online social networks automatically and can infect them in different ways depending on the type of existing vulnerability.

Samy is an example of an XSS worm that spread in *MySpace* in 2005. This XSS worm, which was written in JavaScript, was able to generate one million friend requests in less than 20 h, which finally resulted in a MySpace shutdown for almost 2 h. Another example of an XSS attack was *Mikeyy*, which targeted Twitter in 2009. Mikeyy was an XSS worm that sent out almost 10,000 automated tweets exploiting security vulnerabilities in *Twitter* [7].

[1] http://mashable.com/2009/12/21/facebook-clickjacking/ [Accessed on September 26, 2014]

10.3.7 Cyberbullying

Cyberbullying is a type of attack that takes place by sending out harmful or offensive materials, including text and images, to targeted users. Once such material spreads among a large group of OSN users, it is very difficult, if possible at all, to remove them from the network. Cyberbullying materials are often posted anonymously; therefore tracking their source is not always an easy task. The negative consequences of cyberbullying attacks can be devastating for the target users. They may feel annoyed, unsafe, angry, or humiliated. The negative effects of cyberbullying are usually much more serious on children and youths and typically persist through adulthood. According to a 2013 survey by the *National Center for Educational Statistics* (NCES), almost 30 % of the students reported that they have been the victims of cyberbullying.

10.3.8 Internet Fraud

Due to the huge number of users, investors can use OSNs as a source of information for making financial decisions [3]. At the same time, malicious users can leverage different types of vulnerabilities and weaknesses to manipulate such information and to commit fraud aimed at gaining some kind of monetary benefits. Threats include *nondelivery of merchandise*, *identity theft*, and *credit card fraud*.

For instance, *purchase fraud* is a type of Internet fraud in which a malicious user makes a purchase from a merchant and uses a counterfeit or stolen credit card to make the payment. As a result, the merchant will lose some money because of accepting the counterfeit/stolen credit card and receiving a chargeback as a result.

Using *rogue security software*, also called *Rogue-AV*, is another common method for initiating Internet fraud. Using Rogue Antivirus, the attackers try to deceive a user to download the program in order to remove malware, but it is used as a tool to download malware to the infected systems instead. After being installed, the software is used to induce the user to make an online purchase by displaying enticing messages like offers to solve performance problems or remove malware.

10.3.9 Data Mining and Inference Attacks

Data mining is a powerful tool in the hands of researchers to discover valuable knowledge from large volumes of data. Although information gathered from OSNs using data mining techniques can be a useful resource to evaluate online social networks and improve the quality of services, it can also be used by attackers to extract knowledge that may compromise users' privacy.

An *inference attack* is an example of a data mining technique used by attackers to collect sensitive information from a database by analyzing relatively insignificant data. The main difference between inference attacks and other types of attacks that are mentioned above is that this is not the result of unprivileged access to the data. Indeed, it is an attempt to gather sensitive information from authorized available data. In other words, it is the nature of the information itself that makes it vulnerable to inference attacks [9].

Attackers can take advantage of the information disclosed by OSN users to derive additional sensitive data about them. Using learning algorithms, attackers can even collect some information that is supposed to be private, such as a user's list of friends, preferences and interests, or political views [9].

10.3.10 Sybil and Identity Clone Attacks

Sybil and *identity clone* attacks are similar, as in both types of attacks, the attacker creates multiple unique profiles and uses them to perform malicious activities. In a *sybil attack*, the attacker creates numerous profiles to start an attack against OSN users. In an *identity clone* attack, the malicious user *clones* the victim's profile in order to mislead his friends to trust the duplicated profile so that they would share their personal information with him. Using this mechanism, the attacker can steal the identity and personal information of the victim's friends. In an identity clone attack, the attacker must have some prior knowledge about the target identity [11].

10.4 Available Countermeasures

Although *cryptography* is a strong tool to maintain privacy of data in distributed systems, traditional cryptographic solutions like *Public Key Cryptography* may not be very useful to protect the privacy of users in OSNs since hiding users' identity does not make sense in online social networks. In fact, users must be able to identify other users before they can give them access to their profiles.

Three main issues related to OSN security are *privacy*, *integrity*, and *availability* [4]. With respect to *privacy*, we not only consider securing personal information and content that users share in their profiles, but we also want to keep the communication channels used by OSN members secure against internal or external attacks in order to prevent malicious users from accessing or tampering with data being transferred through those channels. Additionally, we want to prevent any third parties from discovering which users are communicating. Finally, access control mechanisms must be in place to make content available to users who have been granted permission to access such content.

Several vendors—including CheckPoint, Websense, Avira, McAfee, AVG, and Symantec—have begun to develop solutions against different types of attacks in online social networking environments [7]. For instance, Symantec's Norton Safe Web (NSW) provides a Facebook application to identify suspicious websites and insecure links by analyzing users' feeds. Websense Defensio is another Facebook security software which is used to improve security of OSN users by detecting and stopping harmful URLs and scripts.

In the following subsections, we discuss countermeasures available to prevent or mitigate some of the most common classes of attacks against OSNs.

10.4.1 Countermeasures Against Phishing Attacks

There are several factors that contribute to the success of a phishing attack. The main issue is that users may not be able to easily and accurately verify the identity of the sender of email messages. Another problem is that users cannot always differentiate between legitimate and illegitimate contents correctly. It is also possible that the user is not familiar with the meaning of the Secure Sockets Layer (SSL) lock icon, SSL certificates, absence of security indicators, or the difference between genuine security indicators and fake indicators. Additionally, users usually do not pay enough attention to the location bar at every transaction, and they may not be knowledgeable about the structure of domain names and URLs. In brief, a phishing attack is typically the result of the user's reliance on a particular website, logo, and any other trust indicators [5].

There are different classes of *phishing* attacks—including *malware attacks*, *deceptive attacks*, and *DNS-based attacks (pharming)*—whose common purpose is to steal confidential information from the users. On the other hand, there are several steps that can be taken to equip the users against phishing attack.

One solution is to use *signature-based anti-spam filters* that are able to identify *phishing* messages and block them before the users access them. Anti-phishing toolbars and browser plug-ins, such as *Netcraft* or *SpoofStick*, are used to warn users about phishing sites. *Message authentication* is an alternative solution against phishing attacks that provides assurance to users that messages sent to them are from trusted parties. *Personalized visual information* is used as a technique to reduce the likelihood of phishing attacks. An example of this method is using *personalized images* to transfer online messages or selecting a secret image to log into a website. By applying this technique, attackers cannot send deceptive emails since they do not know what personalized information the target user has chosen. For instance, if the secret image is not displayed to the user while logging into his account, it means that he is not on the trusted website and he should not enter his confidential information.

Another countermeasure against phishing attacks is to draw the user's attention to the fact that a message may contain misleading information by using a clear language to explain where a certain link may lead. For instance, if a webpage contains deceptive links, the content will be rendered or highlighted in such a way that the user can visually realize that the embedded URL is suspicious and points to a page that could be malicious.

Banks and other organizations must protect their users against phishing attacks. Using website authentication techniques can assist users in finding out whether they are on a trusted or fraudulent site. In addition, they must timely renew their domain names—to prevent malicious users from taking over expired domains—and regularly search for similar domain names that malicious users may use for phishing attacks. Additionally, it is critical to invest on user education so that users may be able to distinguish between legitimate and fraudulent sites.

10.4.2 Countermeasure Against Sybil Attacks

As has been described earlier in this chapter, *sybil attacks* are based on creating multiple identities by leveraging the open membership characteristic of OSNs. A classical solution against sybil attacks is to use trusted identities endorsed by certified authorities. Although this solution may be effective in mitigating the risk of sybil attacks, it may not be practical to implement in OSNs due to their open membership feature.

Another solution against sybil attacks is to leverage one of the main characteristics of OSNs, which is the trust level built among existing users. Indeed, when looking for suspicious nodes, we can look for members of the network that have fewer connections with others, based on the assumption that sybil nodes cannot establish a large number of connections with other members. On the other hand, nodes that are strongly connected to *non-sybil nodes* can be assumed to be more trustworthy. This *sybil defense scheme* is similar to the graph-partitioning algorithm, where, in this case, the graph represents the OSN, and the objective is to divide the graph nodes into two sets of *trusted* and *untrusted* nodes. However, this approach has some drawbacks. The main issue is related to networks in which it is hard to differentiate between *sybil nodes* and *non-sybil nodes*. In other words, the network may include some communities that are not well connected to each other, causing misclassification of nodes based on the assumptions mentioned above [16].

10.4.3 Countermeasures Against Spamming

Several solutions have been proposed to address *spamming*. One of the most common techniques to detect *spam* is to use *statistical* or *keyword* filtering of messages. In *keyword spam filtering*, the filter looks for suspicious words in the messages using a list of criteria to determine whether the message is spam. On the other hand, *statistical spam filters*—like Bayesian spam filters—compute statistics on how many times tokens, which can be words or other elements of a message, appear in both spam and non-spam messages and then calculate a *statistical probability* to decide if an email is *spam* or not by looking at the tokens in it.

Video spamming is also common in video sharing social networks like YouTube [2]. In this case, a malicious user responds to videos posted by legitimate users with unrelated videos with the sole purpose of advertising products or services, distributing pornography, etc. The techniques used to identify spam in text messages cannot be easily used for *video spam*. Additionally, users must watch at least a part of the posted video to find out if it is spam or not, which will waste system bandwidth and other resources. Some algorithms based on *machine learning* have been proposed to detect spam in online social networks. For instance, Benevenuto et al. suggested an algorithm to detect *video spammers* (rather than *video spams*) by evaluating users' profiles, social behavior, and posted videos [2].

10.5 The Role of OSN Users

As explained in previous sections, users of OSNs deal with various types of privacy and security risks. In this section, we offer simple guidelines that can help OSN users to enhance security and privacy and protect themselves against different types of attacks:

- Users must not share too much personal information in OSNs. Sharing unnecessary private information within a large network can provide malicious users with opportunities to gather or infer personal information about OSN users, putting their privacy and security at risk [7].
- Users must not take the risk of accepting friend requests from unknown people, since such requests are likely to come from malicious users.
- Reading the *Terms of Use* and *Privacy Policies* of the online social network is recommended to users before registration.
- Since the default privacy settings of OSNs are often inadequate, users are advised to modify their settings after joining an OSN so that the information they share in their profile is not visible to unknown people. For instance, *friends only* is typically the best option among available levels of privacy settings and permits only friends of the user to gain access to the information shared in a user's profile.
- Installing *Internet security software* is recommended to protect users' personal information while surfing through OSNs. Another suggestion is to remove unnecessary third-party applications that can potentially gather personal information about the users.
- OSN users must be cautious about *location-based* applications provided by social networks since they can reveal a user's location and trace any movement. Also, it is a good practice that users do not share their contact information, like email addresses, schedules, and routines with others, which might allow malicious users to stalk them.
- Since children are more vulnerable to computer crimes, their parents must monitor their online activities. They must also educate their children about the

inherent dangers of cyber crimes and teach them the basic rules to follow while surfing through the Internet in general and OSNs in particular.

- Users must report any concern they might have about their privacy and security, like *spam*, *cyberbullying*, or *identity theft*. They should consider contacting the OSN provider and local enforcement agencies or consulting knowledgeable attorneys if they think that they are the victims of cyber crime.

In summary, users must be aware of the fact that once their personal information is disclosed online, there is no guarantee that this information can be removed, since it may have been collected by search engines or copied by other users.

10.6 Legal and Regulatory Landscape

As the number of people using online social networks increases, OSNs are becoming a prime target for cyber criminals. Therefore, several laws and regulations have been introduced to protect OSN users from malicious users trying to take advantage of existing weaknesses and vulnerabilities. For instance, some laws exist against *identity theft*. Any activity aimed at collecting personal information with the goal of assuming someone else's identity is considered identity theft, including but not limited to:

- Collecting someone else's identity information or photo to create a fake online social account
- Logging into someone else's email or online social account without his permission
- Deceiving someone to release key personal information like his credit card number using fake emails and websites

In February 2005, US Senator Patrick Leahy introduced two laws to impose tough penalties on people who were convicted of phishing attacks. These two laws have been introduced to disallow the creation of illegitimate websites or procurement of emails that are mainly created for *identity theft* and unlawful access to personal information. Additionally, the *Anti-Phishing Act of 2005*, which was proposed by Senator Leahy, would allow a 5-year jail term and a fine of up to $250,000 for individuals committing phishing attacks. Also, while previous laws only allowed for the prosecution of emails after the fraudulent acts had happened, these two laws allow counterfeit emails to be tracked even if no illegal activity has occurred yet.

However, sometimes it is difficult to track the offenders since most phishing attacks are run by individuals residing in foreign countries. Another concern is that phishing websites usually exist for a short period of time. Also, *phishers* are becoming smarter and tracking them is getting much more complicated than before, if not impossible. Therefore, prosecutors have begun to track any unusual activity that is taking place online even before any financial fraud is reported.

There are also several centers and organizations that have been established to combat cyber attacks. For instance, the *Department of Homeland Security* in cooperation with public and private partners has established the *Multi-State Information Sharing and Analysis Center (MS-ISAC)* and the *National Association of State Chief Information Officers (NASCIO)* to enhance public awareness about cyber security threats.[2] MS-ISAC was established for identification and mitigation of cyber threat vulnerabilities and incident response, whereas NASCIO's main focus is to find advanced solutions for public sector IT challenges and cyber security attacks.

The *US Secret Service* has established a network of *Electronic Crimes Task Forces (ECTFs)* to track and capture malicious activities nationwide. Additionally, the Secret Service has established the *National Computer Forensics Institute* to provide law enforcement agencies with the resources to fight cyber crimes.[3]

The Department of Justice's *Computer Crime and Intellectual Property Section (CCIPS)*—which is working with private sectors, institutions, and other government agencies and foreign counterparts—is another organization that is responsible for fighting cyber crimes. Specifically, CCIPS is responsible for protecting intellectual property (IP)—including any material protected under *copyright*, *trade secret*, or *trademark* laws—against cyber attacks.[4]

The *Federal Bureau of Investigation (FBI)* is another government agency that is in charge of dealing with cyber crimes, including cyber-based terrorism and cyber frauds. Indeed, they usually take the first steps to gather information about fraudulent activities and share it with other agencies worldwide.

Last but not least, the *International Criminal Police Organization (Interpol)* is taking steps against cyber crimes since most computer crimes take place transnationally. Malicious users are now hiring individuals from other countries without diplomatic ties to accomplish their fraudulent activities, including identity theft, phishing, and scamming, with very little chances of being tracked.

10.7 Conclusions

Online social networks have increased disclosure of personal information by making more information available online. Despite all the proactive security monitoring technologies that are used by different online social networks nowadays, cyber attackers still find ways to accomplish malicious activities, like attacking computer systems, engaging in phishing activities and identity theft, cyberbullying, etc.

[2] Defending Against Cybercriminals, http://www.dhs.gov/defending-against-cybercriminals [Accessed on September 26, 2014]

[3] Electronic Crimes Task Forces and Working Groups, http://www.secretservice.gov/ectf.shtml [Accessed on September 26, 2014]

[4] http://www.justice.gov/criminal/cybercrime/ [Accessed on September 26, 2014]

Additionally, attacks against online social networks usually spread faster than other types of online attacks because of the trust existing among the users of the network.

In this chapter, we have reviewed the most common attacks to OSNs, including identity theft, phishing, malware attacks, and sybil attacks. We have also discussed some countermeasures that can be used against these attacks in order to maintain *privacy*, *integrity*, and *availability* of data that is being shared by the users in their online profiles. However, taking everything into consideration, users have a critical role in protecting their own information by adhering to a set of security guidelines. Indeed, users themselves are responsible for any content that they share in their profiles, ignoring the fact that malicious users may find a way to access such content and initiate undesirable activities against them.

References

1. Ashford, W.: RSA discloses phishing-attack data breach details. Online at http://www.computerweekly.com/news/1280095593/RSA-discloses-phishing-attack-data-breach-details (2011)
2. Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., Zhang, C., Ross, K.: Identifying video spammers in online social networks. In: Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web, pp. 45–52 (2008)
3. Brenner, J., Smith, A.: 72% of online adults are social networking site users. Online at http://www.pewinternet.org/ (2013)
4. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy-preserving online social network leveraging on real-life trust. IEEE Commun. Mag. **47**(12), 94–101 (2009)
5. Dhamija, R., Tygar, J.D.: The battle against phishing: dynamic security skins. In: Proceedings of the Symposium on Usable Privacy and Security (SOUPS 2005), Pittsburgh, pp. 77–88 (2005)
6. Faghani, M.R., Saidi, H.: Malware propagation in online social networks. In: Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE 2009), Montreal, pp. 8–14 (2009)
7. Fire, M., Goldschmidt, R., Elovici, Y.: Online social networks: threats and solutions. IEEE Commun. Surv. Tutorials. **16**(4), 2019–2036 (2014)
8. Gross, R., Acquisti, A.: Information revelation and privacy in online social networks. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, pp. 71–80, (2005)
9. Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: Preventing private information inference attacks on social networks. IEEE Trans. Knowl. Data Eng. **25**(8), 1849–1862 (2013)
10. Irani, D., Balduzzi, M., Balzarotti, D., Kirda, E., Pu, C.: Reverse social engineering attacks in online social networks. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Lecture Notes in Computer Science, vol. 6739, pp. 55–74. Springer, New York (2011)
11. Jin, L., Long, X., Takabi, H., Joshi, J.B.: Sybil attacks vs identity clone attacks in online social networks. In: Proceedings of the 6th International Conference on Information Security and Assurance (ISA 2012), Shanghai, pp. 125–127 (2012)
12. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, pp. 29–42 (2007)
13. Potharaju, R., Carbunar, B., Nita-Rotaru, C.: 3-clique attacks in online social networks. Tech. Rep. CERIAS TR 2011-08, Purdue University (2011)
14. Statista - The Statistic Portal: Number of monthly active Facebook users worldwide. Online at http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/ (2014). Accessed April 2014

15. TrendLabs APT Research Team: Spear-phishing email: most favored APT attack bait. Online at http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf (2012)
16. Viswanath, B., Post, A., Gummadi, K.P., Mislove, A.: An analysis of social network-based sybil defenses. ACM SIGCOMM Comput. Commun. Rev. **41**(4), 363–374 (2011)
17. Zhang, C., Sun, J., Zhu, X., Fang, Y.: Privacy and security for online social networks: challenges and opportunities. IEEE Netw. **24**, pp. 13–18 (2010)

Part IV
Context Awareness and Personalization

Automatic system adaptation to a continuously evolving scenario is a key feature of a full-fledged pervasive system, whose reactions should be guided, at all times, by the tastes and needs of the users. This part addresses the main methods adopted to solve this problem, that is, personalization and context awareness. Personalization is devoted to delivering to the users the knowledge that is most relevant for them. Since, in a pervasive system, different information and services may be differently relevant to the same user according to different ambient conditions, the notion of relevance acquires a dynamic flavor; as a consequence, personalization must take into account the different contexts in which users might find themselves.

Chapter 11 introduces the main issues related to personalization, that is, the construction of a user model that will be adopted by the system to adapt its behavior to the user's tastes and circumstances. User models that have been proposed differ mainly depending on the goal of the personalization (presentation, interaction, content reduction, etc.) and on the features of the user that are interesting for the specific application (e.g., preferences, goals, traits, learning styles). A thorough analysis of the main techniques and methods adopted in the literature, and in real systems, will allow the readers to choose the most appropriate method for their specific purpose.

Context, as defined by Dey et al., is represented by the information that can be used to characterize the situation of any entity that is relevant for the system under consideration. Accordingly, context-aware applications and infrastructures support the users to weave their experience in their environment to give it meaning. Chapter 12 introduces the notion of context as intended in the various context-aware systems and proposes a methodology and a model supporting the design of a system delivering context-aware information and services. Moreover, some problems related to the maintenance and evolution of context-aware systems and to context-aware user preferences are introduced.

The ubiquity of mobile devices (e.g., smartphones and GPS devices) has in part motivated the use of contextual information in modern mobile applications. From one perspective, context in mobile systems can fall into three categories: (a) user context that includes the personal attributes of the user, e.g., spatial location and budget; (b) point of interest context, e.g., restaurant location, operating time, and rating; and (c) environmental context, e.g., weather and road conditions. Incorporating such context in applications provided to mobile users may significantly enhance the quality of service in terms of finding more related answers. Chapter 13 first gives a brief overview of context and context awareness in mobile systems. It then discusses different ways of expressing the spatial location context within mobile services. The chapter later describes application examples that can take advantage of various mobile context, namely, social news feed, microblogging (e.g., Twitter), and recommendation and preference services.

A topic in-between Parts IV and V, Chap. 16 presents the use of recommender systems for personalized multimedia selection and presentation.

Chapter 11
Data Personalization

Georgia Koutrika

11.1 Introduction

Personalization has come about as a result of a long evolutionary process accelerated by the rapid development of the Web and the dominance of handheld electronic devices. The Web has enabled people with varied goals and characteristics to access an ever-growing amount of information and online services. Handheld electronic devices, such as tablets and cellular phones, have made information access possible from anywhere and anytime. At the same time, the massive data generated by devices, applications, and people creates overload but also enables new, improved applications and services. In this context, personalization offers a meaningful way to sort out choices and present them to users taking into account their individual or group characteristics.

Personalization approaches can be found in several systems, from Web search engines and e-commerce sites to digital libraries and museums. Personalization can be experienced at different levels. At the presentation level, personalization can affect the way the application or service is presented to the user. At the interaction level, the types of interaction may be tailored to the user. Personalization approaches have been developed by different research communities, including information retrieval (IR), data mining, databases, hypertext, and human-computer interaction.

At the core of every personalization effort, there is a user model that dictates how a user is represented at the system level. The personalization method employed prescribes how the system adapts its behavior to the user. A wealth of user models have emerged from different communities that vary depending on the features of the user modeled (e.g., preferences, goals, traits, learning styles), the application context

G. Koutrika (✉)
HP Labs, Palo Alto, CA, USA
e-mail: koutrika@hp.com

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_11

(e.g., search, e-commerce), and the purpose of the personalization (personalization of content, interaction, etc.).

The purpose of this chapter is to approach personalization in the context of pervasive data management and under the prism of applications such as smart cities, cultural heritage sites (e.g., indoor museums, archaeological sites, historical archives), and so forth. Hence, we will focus on aspects of personalization that are most relevant in this context. At the same time, we will also include an outline of basic and related material in order to provide a self-contained presentation of the related personalization efforts. As our guiding example, we consider a tourist who desires to visit a museum that offers exhibitions of pictures and sculptures. We will review user modeling and personalization approaches that are pertinent in this context.

11.2 Definition of Personalization

Personalization can be broadly defined as providing an overall customized, individualized user experience by taking into account the needs, preferences, and characteristics of a user (or group of users).

The focus of this chapter is on data personalization. Given a collection of items, *data personalization* provides different views of the collection to different users. Depending on the application, items can be products, movies, books, etc. For our museum example, items can be artifacts, exhibitions, rooms, etc. A data personalization method may perform the following operations:

- *Reorder* the items in a collection to be shown to a user. For example, in our museum scenario, the artifacts or the museum exhibitions may be ordered in different ways to be explored by different tourists. As a simple example, a person may prefer chronological ordering, while another may prefer ordering based on artifact popularity.
- *Focus* on the items of interest. For example, people with different interests may be shown different options in the museum, such as classical period artifacts vs. Greek-Roman period artifacts.
- *Recommend* additional options or suggestions. For example, if a tourist liked a particular artifact of a painter, additional artifacts of similar painters may be provided for exploration.

Different research communities, including information retrieval, databases, and machine learning, have contributed a variety of personalization approaches. These approaches attack personalization from different perspectives. For example, information retrieval approaches for data personalization have mainly focused on finding ways to improve the results of unstructured searches by taking into account information about the users. On the other hand, database-driven personalization efforts have focused on user preference modeling for structured data and algorithms for efficient computation of queries with preferences. Machine learning and data mining methods have been applied for the development of recommendation strategies. Most approaches are data dependent. For example, user preference models for structured data are different from models for unstructured data. Several approaches are application driven, such as methods that have been particularly developed for adaptive educational systems.

In this chapter, we will not cover such application-dependent approaches. We divide personalization methods into three large categories depending on the personalization effect: information filtering, personalized search, and recommendations. For each category, we will provide several examples of how these can be seen in the context of a data pervasive application, using our museum example. In the following section, we will first describe the various types of information that can comprise a user profile and can be used by a personalization method.

11.3 User Models and Profiles

A *user model* is a representation of a user. According to the nature of the information that is being modeled, we can distinguish three broad types of user models:

1. *User feature models* represent features of the user as an individual such as demographic data, needs, preferences, goals, as well as physical, cognitive, and behavioral characteristics.
2. *User access models* focus on processes, actions, as well as rights that enable the user to accomplish tasks.
3. *User context models* represent the current context of the user's work.

User feature models are important to all adaptive systems. There is a plethora of systems that use different types of user models for different types of applications, such as online recommendation and e-learning systems (e.g., [10]). There are also application-independent models emerging from diverse fields such as psychology, artificial intelligence, information retrieval [22], and databases [63] that model human preferences and behavior.

User access models are typically met in operating systems, file systems, and database systems, but they can also be part of any adaptive system. For example, tourists who visit a special exhibition may have different privileges depending on their status or profession. Finally, user context models are mostly the concern of mobile and ubiquitous adaptive systems, where context is essential, and they are explained in different chapters of this book [52, 58].

A *user profile* is an instantiation of a user model representing either a specific real user or a representative of a group of real users.

This chapter focuses on user feature models. In particular, we focus on the following most common user features:

- *Knowledge*: what the user already knows
- *Goals*: what the user wants to achieve
- *Preferences*: what the user is interested in

Table 11.1 User modeling approaches

User knowledge	Adaptive educational systems [10, 32, 65]
User goals	Intelligent information agents [2, 14, 17, 41, 42, 56]
	Goal-enhanced query answering [7, 9, 26, 57]
User preferences	Recommender systems [3, 5, 30, 43–45, 55, 60, 62, 66]
	Personalized search [31, 40, 51, 61]
	Database query processing and personalization [1, 16, 36, 39, 63]

In practice, not all of these user modeling questions are pertinent to all applications. For example, in e-commerce, user knowledge is not a factor that determines user purchase behavior. On the other hand, in an e-learning scenario, user preferences or interests do not matter as much as a user's knowledge and goals. Students may pursue a good grade in a "history" course irrespective of their liking of the subject.

Let us consider our museum example. Knowing what the user knows could help a personalization system decide how much information to show a user. For example, if the user does not know about a certain period, then the system may decide to provide an introduction to this time period before going through particular artifacts of the period. Knowing the goal of the visit(s) of a person, a system can tailor the visit to meet this goal. For example, the goal may be "to have a 2-hour tour." Finally, knowing the user preferences can help suggest artifacts of interest to this person.

Table 11.1 summarizes example efforts on modeling knowledge, goals, and preferences. These models are presented in the subsequent sections. The table groups user models depending on the context or the application they are designed for, such as database query processing or personalized search.

11.3.1 User Knowledge

The user's knowledge on the subject matter or the domain is an important factor that can determine how much information and in what depth the user should be exposed to. Returning to our museum tourist example, domain experts that visit the exhibition museum may want to find out more details on each of the exhibit's artifacts, while a person with little knowledge on the field may need a more concise and easy-to-understand summary of the main artifacts.

The user's knowledge can both increase (learning) and decrease (forgetting) from session to session and even within the same session. This means that an adaptive system relying on user knowledge has to recognize the changes in the user's knowledge state and update the user model accordingly [10]. Representations of user knowledge have been studied in adaptive educational systems (e.g., [32, 65]).

The simplest way to model user knowledge is to use a *scalar model*, which maps the level of user domain knowledge to a single value on some scale—quantitative (e.g., a number ranging from 0 to 5) or qualitative (e.g., good, average, poor, none). Scalar models are typically produced by user self-evaluation or objective testing. For example, a tourist entering the exhibit may choose a self-characterization, such as expert or newbie.

Simple scalar models aggregate user knowledge into one value. On the other hand, *structural models* assume that domain knowledge can be divided into certain independent fragments or aspects, and they represent user knowledge of different fragments independently. The most popular form of a structural knowledge model is an overlay model [65]. The purpose of the overlay model is to represent an individual user's knowledge as a subset of the domain model, which reflects the expert-level knowledge of the subject. For each fragment of domain knowledge, an overlay model stores some estimation of the user's knowledge level of this fragment. The pure overlay model, developed in the field of educational systems, assigns a Boolean value, yes or no, to each fragment, indicating whether the user knows or does not know this fragment.

11.3.2 User Goals

The user's goals represent the immediate purpose for a user's work within an adaptive system. Depending on the kind of system, it can be the goal of the work (in application systems), an immediate information need (in information access systems), or a learning goal (in educational systems). In all of these cases, the goal is an answer to the question "what does the user want to achieve?"

A large part of research on adaptation to the user goals was done in the area of intelligent information agents. Emerging from the intersection of Artificial Intelligence (AI) and human-computer interaction, *intelligent interfaces* can be incorporated into a normal application such as a Web browser, a text editor, or a search engine and are able to perform all the operations that the user is allowed to perform through the interface [2, 41]. Although such agents do not perform data personalization, we mention them here as they are applicable to the type of applications we focus on in this book. For example, when the user is buying the tickets for a museum visit, an intelligent interface may help the user manage and expedite user accounting and registration.

Another category of Artificial Intelligence agents that consider user goals is story generators. *Story generators* generate stories given a description of a hypothetic environment [14, 56]. Story generators can be implanted in applications in the field of entertainment, training, or education. For example, tourists visiting a museum may have different goals, such as "to learn about the important artifacts" or "to learn about the classical period." Story generators may be used to generate narratives for satisfying different user goals. However, these systems are clearly at an early stage of development. In their current state, they are obviously not producing the depth and richness of narrative. Hence, the story is identified as the optimal sequence of actions that results in the fulfillment of a goal of a story character.

Finally, *dialogue systems* are computer systems intended to converse with a human with a coherent structure and help achieve their goals. Dialogue systems can support a broad range of applications in education, healthcare, and entertainment, such as responding to customers' questions about products and services and informational services about news, entertainment topics, the weather, or any other type of information stored in a knowledge base [17, 42].

In the systems mentioned above, goal representation approaches include plan libraries, consistency graphs, Markov models, and Bayesian networks [48]. Although these models have been applied to problems other than data personalization, it is worth noting that they are by far the most sophisticated ones to goal representation, since they represent not only different user goals but also ways (plans) for achieving these goals.

Adaptation to the user's immediate information need has also been explored by adaptive information retrieval systems [7]. Web information retrieval originally deals with the retrieval of information that might be useful or relevant to a user query. In the last two decades, there have been efforts to enhance search engines by additionally considering what the user wants to do after having access to the retrieved information, in other words, by considering user goals [9, 26].

Goal-enhanced information retrieval methods can offer to the user something more than plain information. They can provide direct access to services such as "take a short tour" and "explore museum in 2 hours." The answering of a user query based on a goal that has been inferred from the current query or a sequence of antecedent user queries is called *goal-enhanced query answering*.

Goal taxonomies of queries in the context of Web information retrieval were derived from extended user studies using questionnaires and interactive tools on Web browsers tracking user moves such as clicks and form submissions in combination with expert knowledge [9, 57]. To come up with the categorization, experts actually studied which are the user intentions (otherwise called information needs) that drive users to search on the Web. Examples of user intentions are "to be informed," "to navigate to a site," "to execute a transaction," and "to get advice."

For example, the taxonomy that Broder suggested consists of three categories of Web queries [9]: (1) informational queries, which consist of terms that describe or capture vague notions or consist of specialized terms that indicate that the desired target is to enlighten the user on the subject; (2) navigational queries, which consist of query terms that describe a specific URL; and (3) transactional queries, which contain terms that indicate that the target URL enables a transaction such as downloading a file, buying an item, or watching a video.

In all the systems above, inferring user goals is very challenging. One possibility is that the system has a set of predefined goals and the user explicitly chooses one of them. For example, museum visitors could choose among goals such as "to learn about the important artifacts" or "to learn about the classical period." Then, the system could offer different types of information for each goal. Implicit user goal inference is much more tricky. The system needs to observe the user, recognize the user goal based on the observations, and adapt as the goal may change. For example, based on how much time a visitor spends for each artifact, the system could infer whether the user goal is to have a short or longer visit in the museum.

11.3.3 User Preferences and Interests

Preferences have been traditionally studied in philosophy, psychology, and economics and applied to decision-making problems. In the context of computer systems, user interests always constituted the most important (and typically the only) part of the user profile in adaptive information retrieval and filtering systems that deal with large volumes of information. It is also the focus of user models/profiles in recommender systems and databases. More recently, the abundance of available content and the growing popularity of the interest-driven constructionist approach to education have encouraged more attempts to model user interests in educational systems as well [10].

How user preferences are modeled largely depends on the type of data and application. Hereafter, we distinguish two large categories of user preference models: over unstructured and structured data.

11.3.3.1 Preference Models for Unstructured Data

Information filtering, recommendation, and personalized search systems are based on some kind of user profile. User profiles fall into the following categories.

Unstructured information retrieval models. A user is represented in the same way as documents. In this case, classical models from information retrieval that are used for document representation are applied for profile representation as well. A user profile may be represented as a bag of words [30]. No distinction is made among more and less important terms. Another possibility is that a user profile stores one or more vectors of terms. The weight of a term indicates the user interest in this term and ranges from -1 to 1 [3, 66].

Networks of terms. These are graphs that consist of nodes representing terms and edges representing relationships among terms. A term may be associated with a weight in the range [0, 1] showing user interest (PSUN [62], ifWeb [44]). An edge may also carry weight showing the significance of the corresponding relationship. A relationship may correspond to co-occurrence [62] and semantic or syntactic relationships (semantic nets [44], ontologies [43], concept hierarchies [51, 61]).

The concept overlay approach to user interest modeling is very similar to the overlay knowledge modeling approach. Concept-level models of user interests are generally more powerful than keyword-level models. Concept-level models allow a more accurate representation of interests. Given a rich domain model, a concept overlay model can separately model different aspects of user interests. For example, a news personalization system can model user interests on distinct topics, based on a specific geographical location, dealing with specific named entities [31]. An adaptive museum system can separately model interests in the designer, style, or origin of a jewelry item [46].

Machine learning models. When machine learning methods are employed for user profiling, the structure used by the method (e.g., a neural network [5], decision tree [40], or Bayesian network [45]) comprises the user profile.

A neural network is a network of input and output cells, based upon neuron functions in the brain. Neural networks create a compact representation that responds to queries quickly. However, they can be slow to train. For example, Re:Agent [5] filtered e-mails through a neural network previously trained with feature vectors of past messages.

A decision tree is another way to classify data. It consists of a set of nodes and a set of directed edges that connect the nodes (tree structure). The internal nodes represent questions about the parameters, and the edges represent answers to those questions, i.e., values for the parameters. The leaf nodes represent a final decision. For example, InfoFinder [40] recommends documents based on a decision tree.

A Bayesian network is a directed acyclic graph in which nodes represent propositional variables and arcs represent dependencies. A node value is a function of the values of the nodes it depends upon. Leaf nodes represent propositions, which can be determined by observation. The resulting model is very small, very fast, and accurate [8].

Association rules have been used for many years in merchandizing, both to analyze patterns of preference across products and to recommend products to consumers based on other products they have selected. Association rules can form a compact representation of preference data which improve efficiency of storage as well as performance. For example, an association rule expresses the relationship that a certain movie is often purchased along with others [4].

User-rating matrix. Collaborative filtering systems do not store a separate profile for each user, but a matrix. The rows of this matrix correspond to users, the columns to items, and each cell the rating, if any, a user has given to a particular item [55, 60].

11.3.3.2 Preference Models for Structured Data

In databases, interest in preferences was triggered by observing the limitations of the Boolean database answer model, where query criteria are considered as hard by default and a nonempty answer is returned only if it satisfies all the query criteria.

Preferences are used in query processing to provide users with customized results typically through ranking. They have also been used in query personalization, i.e., dynamically modifying a user query to include preferences [39]. The results of personalized queries may be smaller than the results of the original query and are ranked according to user preferences.

There are two different lines of work on using preferences in query processing [63]. In general, preferences can be expressed either qualitatively or quantitatively.

In the *qualitative* approach, preferences between database tuples are specified directly, typically using binary preference relations. Preference relations may be specified using logical formulas [16] or special preference constructors [36].

In the *quantitative* approach, preferences are expressed by assigning numerical scores to database tuples. In this case, a tuple is preferred over another tuple, if and only if its score is higher. Scores may be assigned through preference functions (e.g., [1]) or as degrees of interest associated with specific conditions that must be satisfied (e.g., [39]). These methods assume the existence of a number of user preferences and appropriately rewrite regular database queries to incorporate them. This process is often referred to as *query personalization*.

11.4 Data Personalization Methods

In this part of the chapter, we present data personalization methods organized into three categories: information filtering, recommender systems, and personalized search. Table 11.2 provides several example efforts in each category.

11.4.1 Information Filtering

Information filtering systems filter out irrelevant information from incoming information streams or collect and distribute relevant information based on some profile [20, 45]. Filtering of information has never been a new concept nor is it one that is limited to electronic documents. We apply filtering in our everyday tasks when we buy certain magazines and read certain columns of newspapers. Storing information in electronic form has created the possibility that some of this filtering is done automatically by a system.

Information filtering and information retrieval have been called the two sides of the same coin. However, information retrieval is about comparing one query to a

Table 11.2 Personalization approaches

Information filtering	[20, 44, 45, 62, 66]
Recommenders	Content-based recommenders [43, 50]
	Collaborative filtering [8, 15, 18, 21, 23–25, 28, 34, 55]
	Knowledge-based recommenders [12, 13]
Personalized search	Query personalization [38, 39, 51, 61]
	Personalized result ranking [11, 29, 51, 53, 64]

relatively static collection of documents, while in information filtering, an incoming set of objects is compared to many profiles (queries) at the same time.

As an example of an information filtering scenario, a tourist subscribes to the museum news feed. As part of the subscription, she/he may specify categories of information she/he would like to receive updates on, such as new painting exhibitions. This user profile is used by the system to deliver news of interest to this person.

In many systems, filtering is performed by computing the similarity between a filter (user profile) F and a collection, which results in the generation of a list of items ranked based on their similarity to F. The similarity may be computed as the inner product or cosine similarity of the corresponding vectors. A threshold is used to decide which items are shown to the user. This is called *dissemination threshold*. This threshold may actually vary for different people, depending on factors such as the information domain and the cost of retrieving the information.

Filtering systems have been traditionally used for e-mails, news, documents, music, and TV channels [44, 66]. Common user profile representations are borrowed from information retrieval and include the Boolean, vector-space, and inference models. Filters may also be in the form of rules. Rules are more easily defined for (semi-)structured data. Semantic nets have been used as well [62].

For example, SIFT [66] is a Usenet news filtering system that requires users to specify keywords for the initial filter, which may be represented using the Boolean or vector-space model. If the latter is chosen, SIFT can provide some filter adaptability. For this purpose, it requires users to provide relevance feedback by pointing out interesting documents; based on that weights in the profile are adjusted accordingly. A user subscribes to a SIFT server with one or more subscriptions, one for each topic of interest. Apart from the IR-style profile, a subscription includes additional parameters to control the frequency of updates, the amount of information to receive, and the length of subscription. The user specifies a relevance threshold, which is the minimum similarity score that a document needs to have against the profile to be delivered. A default value is provided for convenience.

As another example, LIBRA [44] is a book-recommendation system that utilizes information extraction and a machine learning algorithm for text categorization. It uses a database of book information extracted from Amazon. Users provide 1–10 ratings for a selected set of training books. The system then learns a profile of the user using a Bayesian learning algorithm and produces a ranked list of the most recommended additional titles. LIBRA does not attempt to predict the exact numerical rating of a title but rather a total ordering of titles in order of preference. This task is then recast as a probabilistic categorization problem of predicting the probability that a book would be rated as positive rather than as negative, where a user rating of 1–5 is interpreted as negative and 6–10 as positive. After reviewing recommendations, the user may assign his own rating to examples and retrain the system.

11.4.2 Recommender Systems

A recommender system provides predictions, recommendations, and opinions that help a user in evaluating and selecting objects [54, 59]. The basic idea is that the user selects, in some way, objects, for instance, by clicking on them or querying about them, and the system identifies other similar objects, based on which it produces recommendations or predictions regarding what the user would like. Recommender systems have become popular in many Web sites, such as Amazon, Google News, and Netflix, for recommending products, news, movies, and other topics.

Recommender systems can be classified into two categories: content-based and collaborative filtering. Content-based filtering analyzes the associations between user profiles and the descriptions of items and recommends items similar to those the user liked in the past. Collaborative filtering methods rely only on past user behavior (e.g., transactions or ratings) to make recommendations.

11.4.2.1 Content-Based Approaches

Content-based approaches are derived from concepts introduced by the information retrieval community [43, 50]. The underlying idea is "find me things like those I have liked in the past." User profiles and items of a collection are represented as vectors of terms. Text categorization or classification methods are used for dividing the items into two sets: those recommended and those ignored.

Content-based filtering has the benefit that new items can be easily recommended based on their similarity to users' past choices. However, it suffers from two important disadvantages.

The first one is content limitation. Content-based filtering builds on IR methods, which can only be applied to a few kinds of content, such as text and image, and the extracted features can only capture certain aspects of the content. For example, for Web pages, IR techniques completely ignore aesthetic qualities, multimedia, and network factors. The second limitation of content-based filtering is overspecialization: it constantly provides recommendations merely based on user profiles. Therefore, users have no chance of exploring new items that are not similar to those items included in their profiles.

11.4.2.2 Collaborative Filtering

The term "collaborative filtering" was coined by the developers of Tapestry [24]. Collaborative filtering systems have automated the everyday procedure of relying on recommendations from others whenever personal experience of the alternatives is not sufficient for making choices.

A pure collaborative filtering system is one that performs no analysis of the items at all; instead the only thing it knows about an item is a unique identifier and relies on past user behavior (e.g., transactions or ratings) to make recommendations. Two large categories of collaborative filtering approaches are distinguished: memory-based and model-based ones.

With *memory-based* algorithms, the entire recommendation process is generally an online process. People provide recommendations (e.g., ratings) as input to the system. Then, users similar to the target user are identified (correlation step), and items are recommended based on the preferences of the similar users. Typically, users are considered similar if there is overlap in the items watched or consumed and similar ratings have been given. The opinions of the similar people on each item are aggregated to a score that hopefully reflects the preference of the target user as well (aggregation step). In some cases, the system's value lies in its ability to correlate recommenders and information seekers; in others, the primary transformation is in aggregation.

The nearest-neighbor algorithm is the earliest memory-based technique used in recommendation systems [55]. With this algorithm, the similarity between users is evaluated based on their ratings of items, and the recommendation is generated considering the items visited by nearest neighbors of the user. In its original form, the nearest-neighbor algorithm uses a two-dimensional user-item matrix to represent user profiles. Aggregation is performed as artifact rating or classification of items of the matrix. In order to correlate people, several measures have been used for computing the similarity between users: Pearson Correlation [55], Constrained Pearson Correlation [60], vector similarity [8], and Spearman Correlation [25]. Predictions of how much a user will like an object are computed by taking the weighted average of the opinions of a set of nearest neighbors for that object.

Model-based algorithms use the user database to estimate or learn a model (e.g., Bayesian [15], probabilistic relational [23], and probabilistic Latent Semantic Analysis (LSA) models [28]) that captures item or user relationships. Thus, the time-consuming part of the recommendation process is performed offline, leaving the online process with reasonably low time complexity. During the online operation of the system (prediction), predictions of how much a user will like a new object are computed with the help of this model. From a probabilistic perspective, the collaborative filtering task can be viewed as calculating the expected value of a vote, given what we know about the user. Model-based approaches are preferred in large-scale applications, where millions of recommendations are to be generated in real time.

Bayesian networks (e.g., [8, 15]) create a model based on a training set with a decision tree at each node and edges representing consumer information. The model can be built offline and is very small, fast, and essentially as accurate as nearest neighbor methods. Bayesian networks may prove practical for environments in which knowledge of consumer preferences changes slowly with respect to the time needed to build the model but are not suitable for environments in which consumer preference models must be updated rapidly or frequently.

Classifiers have been quite successful in a variety of domains ranging from the identification of fraud and credit risks in financial transactions to medical diagnosis to intrusion detection. Basu et al. [4] built a hybrid recommender system that mixes collaborative and content filtering using an induction-learning classifier.

On the other hand, clustering techniques identify groups of consumers who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other consumers in that cluster. Some clustering techniques represent each consumer with partial participation in several clusters. The prediction is then an average across the clusters, weighted by the degree of participation.

Clustering techniques usually produce less personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest-neighbor algorithms [8]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller. Clustering techniques can also be applied as a "first step" for shrinking the candidate set in a nearest-neighbor algorithm or for distributing nearest-neighbor computation across several recommender engines. While dividing the population into clusters may hurt the accuracy or recommendations to users near the fringes of their assigned cluster, pre-clustering may be a worthwhile trade-off between accuracy and throughput.

Recent model-based methods use pattern discovery techniques. Patterns that are mined can be association rules, i.e., rules for groups of co-occurring objects, or sequential patterns capturing strict orders of items.

Association rules are useful for recommending items related to a particular item [21, 34]. A classical example is the discovery of sets of products usually purchased together by many independent buyers, which is applied to recommend other products to a shopper related to the product currently being viewed (as in the "Customers who bought this item also bought" recommendations in Amazon).

Association rules do not generate any ordering relation between the items associated. On the other hand, work studying temporality or order in the context of recommendations deals with prediction problems where there is a strict order or path followed by the user, and one wants to predict the next step in that sequence [18, 19]. A typical example is predicting the next Web page a user will access given a sequence of pages visited up to now (the next-page prediction problem [18]).

Let us see some examples of collaborative filtering type recommendations in our museum example. A nearest-neighbor method could be used to find visitors that liked the same artifacts and make recommendations to a visitor based on the opinions of other like-minded visitors. As a source of user preferences, we could use ratings that different visitors gave through the museum social application. As another source, we could use the time spent in front of each artifact as an implicit indication of user interest. Mining association rules could identify artifacts that visitors visit together and make recommendations to a new visitor. Mining the paths followed by visitors could predict the next artifact or room a visitor would like to access given the artifacts already visited.

Collaborative filtering has a number of advantages compared to content-based recommendations:

- It can support filtering items whose content is not easily analyzed by automated processes.
- It can filter items based on features such as taste and quality.

- It can provide serendipitous recommendations, i.e., recommend items that are valuable to the user, but do not contain content that the user has seen or expected.

Collaborative filtering has disadvantages too. To be able to make accurate predictions, the system must first learn the user's preferences from user ratings. If the system does not show quick progress for a new user, the user may lose patience and stop using it. On the other hand, if a new item appears in the database, there is no way it can be recommended until some users rate it. Both these problems are different instances of the *cold-start problem*.

Sparsity is another curse for collaborative filtering. If the number of users is relatively small to the volume of information in the system, then there is a danger of very sparse ratings. Effective generalization from a small number of examples is thus critical. Finally, scalability is important. Predictions need to be made in real time and many predictions may potentially be requested at the same time. The algorithms need to scale well with the number of users and items in the system.

11.4.2.3 Knowledge-Based Recommenders

Knowledge-based recommenders use knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements. An important class of knowledge-based systems draws from research in case-based reasoning [35].

For example, the restaurant recommender Entree [13] makes its recommendations by finding restaurants in a new city similar to restaurants the user knows and likes. The system allows users to navigate by stating their preferences with respect to a given restaurant, thereby refining their search criteria. The FindMe technique is one of knowledge-based similarity retrievals. There are two fundamental retrieval modes: similarity and tweak application.

In the similarity case, the user has selected a given item from the catalog (called the source) and requested other items similar to it. First, a large set of candidate entities is retrieved from the database. This set is sorted based on the similarity to the source and the top few candidates are returned to the user. Tweak application is essentially the same except that the candidate set is filtered prior to sorting to leave only those candidates that satisfy the tweak. For example, if a user responds to an item with the tweak "nicer," the system determines the "niceness" value of that source item and rejects all candidates except those whose value is greater.

Initial FindMe experiments demonstrated something that case-based reasoning researchers have always known, namely, that similarity is not a simple or uniform concept. Therefore, this method leveraged user goals and employed goal-based similarity measures. These measures estimated how close an item is in meeting the user's goals. For example, in the Entree restaurant recommender system, the ranking of goals was cuisine, price, quality, and atmosphere, which seemed to capture our intuition about what was important about restaurants.

As part of the development of WPS [12], it was realized that different users might have different goal orderings or different goals altogether. This gave rise to the concept of the retrieval strategy. A retrieval strategy selects the goals to be used in comparing entities and orders them. For example, for selecting wines, the standard goal ordering would be wine type, price, quality, flavor, body, finish, and drinkability. The system also had a "money no object" strategy in which price is not considered. Given the goal ordering in the strategy, the process of finding the most similar candidate becomes an alphabetic sort. Candidates are sorted into a list of buckets based on their similarity to the source along the most important goal. Then, each of these buckets is sorted based on the next most important goal, creating a new list of finer-grained buckets and so forth.

A knowledge-based recommender system does not have a ramp-up problem since its recommendations do not depend on a base of user ratings. It does not have to gather information about a particular user because its judgments are independent of individual tastes. These characteristics make knowledge-based recommenders not only valuable systems on their own but also highly complementary to other types of recommender systems.

11.4.2.4 Hybrid Recommenders

Hybrid systems combine the advantages of both worlds while avoiding their disadvantages. In the case where both the content features and similarity features are present, the content features can ease the recurring startup problem, while the similarity features can capture some of the subjective qualities of the item not captured by the content features. There are various combination approaches [33].

One approach is collaboration via content. Collaborative methods look for similarities between users. For this purpose, the pattern of ratings of individual users is used to determine similarity. In collaboration via content, the content-based profile of each user is exploited to detect similarities among users. Again, computing Pearson coefficient based on weights stored in the profiles is possible, and the prediction is determined by a weighted average.

Another approach is to combine the output of multiple methods looking for consensus among algorithms. There are several options here: using linear regression to produce a "best fit" combination for a given user or taking the average or the best output or using a voting mechanism.

Fab [3] was among the first systems to combine content analysis and collaborative filtering. It has classes of adaptive content-based collection and selection agents, whose interaction fosters emergent collaborative properties. A collection agent has a search profile. A selection agent corresponds to a user. At regular intervals, collection agents submit the pages they have gathered, which best match their search profiles to the central repository, replacing pages they have previously submitted. Thus at any time, the repository contains each collection agent's best pages. The central router forwards the pages onto those users whose profiles match some threshold above. Additional functionality is located within a selection agent, which among other things collects user feedback. Additionally, any highly rated pages are passed directly to the user's nearest neighbors, i.e., other people with similar profiles. These collaborative recommendations are processed by the receiving user's selection agent in the same way as pages from the central router. In this way, users receive items both when they score high against their own profile and when they are rated highly by a user with a similar profile.

11.4.3 Personalized Search

Traditionally, both databases and information retrieval have been built upon a query-based information access paradigm, i.e., the answer to a query depends only on this query. Therefore, all users issuing the same query are provided with the same answer. This deterministic behavior is desired in order to provide users with the same view of information; however, it often hinders users from locating relevant information due to information abundance and user heterogeneity.

Focusing on the user enables a shift from what is called "consensus relevancy" where the computed relevancy for the entire population is presumed relevant for each user toward "personal relevancy" where relevancy is computed based on each individual's characteristics. Storing user preferences in user profiles gives a system the opportunity to return more focused personalized (and hopefully smaller) answers. The primary ways to personalize a search for an active searcher are query personalization and personalized result ranking.

11.4.3.1 Query Personalization

Query personalization methods consider that user preferences are provided as a user profile. At query time, the goal is to find which preferences should be taken into consideration in the context of a query and integrate them into query processing.

At a high level, query personalization involves two phases: preference selection and personalized query processing. Preference selection determines which preferences from a user profile can possibly affect the query based on how they relate to the query and the external context at query time. For example, assume that a museum visitor wants to explore the painting collection. Then, any preferences for sculptures are not relevant. Personalized query processing involves integrating preferences into the query and returning a preferential (or personalized) answer.

Different approaches on query personalization have emerged from information retrieval and databases, the form of the data (unstructured vs. structured) playing a major role in preference representation, and the algorithms developed in each case.

For example, ARCH [61] performs concept-based query enhancement based on domain-specific concept hierarchies and user profiles together with interactive query formulation. It differs from traditional relevance feedback methods in that it assists users in query modification prior to searching.

First, the user enters a keyword query. Then, the system retrieves relevant portions of the concept hierarchy and displays them to the user. For this purpose, it matches the term vectors representing each node in the hierarchy with the list of keywords typed by the user. Those nodes that exceed a certain threshold are displayed along with adjacent nodes. The user can select and deselect concepts leading to the creation of an enhanced query. The formula for the enhanced query is based on Rocchio's method for relevance feedback.

Database approaches have focused on how to define relevant preferences and efficient query computation. Intuitively, the selected user preferences combined with the query should yield an interesting, nonempty result [38, 39].

11.4.3.2 Personalized Result Ranking

Personalized result ranking can be performed by computing the similarity of results to the user's profile. The initial user query is executed unchanged and results obtained are re-ranked according to their similarity to the user profile (CASPER [53], METIORE [11]).

METIORE allows users to access the bibliographic references of research publications available at the library of the computer laboratory center LORIA. Their personalization approach is based on the concept of objective that is provided by the user when the system is launched. The use of the system for a given objective is during a session. The objective is presently used to group the set of queries, concepts, and decisions that the user makes in the system with the objective in mind. Search queries are defined using the attributes of a publication such as title, authors, year, keywords, etc. METIORE sorts the solution in the order of user preferences, giving the most relevant solution at the beginning.

A different perspective on personalized result ranking is considered in graph analysis. Algorithms such as PageRank [47] and Hubs-Authorities [37] have been applied for ranking results of Web queries. These algorithms measure the importance of each page in the Web by taking into account parameters such as its location and connectivity and rank results to a query not only based on relevance but also based on importance. Recently, modifications of these approaches have been proposed that take into account a user's personal view of page importance [29, 64] to create personalized rankings.

Persona is a system based on ODP hierarchy [64]. The user profile maps a query to nodes in the hierarchy and is used for re-ranking the results of a query. The user inputs a query, which will then be outsourced to an existing search engine, the result of which is filtered to leave the top results. The system tries to find which nodes in the ODP taxonomy these documents correspond to, and it updates the result weights. The results are passed back to the graph-based search algorithm (gradient ascent HITS). Nodes with positive weights are given positive bias, while nodes with negative weights are given negative bias. The system simply filters out the negatively weighted URLs. The user may give positive or negative feedback on the returned set of documents. The system will then refine the current results based on these preferences, giving higher weights to the positively rated pages and lower weights to the negatively rated pages. In addition, this user feedback is incorporated into the user profile.

Outride is another system that builds user profiles upon the ODP hierarchy [51]. It performs both query personalization and personalized result ranking. For the former, the similarity of each query term to the user profile is computed, in order to determine which of the terms stored in the profile may be combined with the query. For example, if a person is interested in coffee and issues a query about "java," then the system may add the term "coffee" in the query. If another user interested in programming searches for "java," the system may augment the query by adding the term "programming" or "language" to provide this user with results about the Java programming language. Result ranking is performed by comparing the results retrieved to the current user profile.

11.5 Personalization Examples in Museums

In the past few years, personalization through information technology has become an increasingly significant trend in the museum world [6], where it has been introduced to make facilities more relevant and useful for individual users. By providing differentiated access to information and services according to the user's profile, personalization helps to respond to museums' educational, marketing, as well as usability needs. Given the growing number of visitors looking for online information concerning their collections and activities, it has become fundamental for these institutions to try to improve their visitors' ability to navigate online and to access information in the most effective way. Personalization is a viable and worthwhile aid in this process.

Personalized access to collections, alerts, agendas, tour proposals, and tour guides is just an example of the different applications that have recently been developed by museums all over the world. Personalization has the advantage of improving the usability of a museum's Web site by facilitating navigation and helping people find the right information and by taking the differences in age, level of education, learning style, and previous knowledge into consideration when providing information. Personalization can also be applied to individualize how each person experiences a visit to the premises of a museum by providing personalized tours, narratives, and exploration paths.

Furthermore, personalized systems have the potential to facilitate the learning process. Visitor studies seem to confirm that learning is stimulated when the information is described in terms that the visitor can understand and if it makes reference to their interests as well as concepts that the visitor has already encountered during the navigation of the museum Web site or an actual visit to the museum.

The first examples of Web personalization in a museum context were developed in the second half of the 1990s in strict relation with the affirmation of academic research on adaptive hypermedia. The ILEX (Intelligent Labelling Explorer) project, for example, was developed between 1996 and 1997 by the University of

Edinburgh in cooperation with the National Museum of Scotland, with the aim of providing intelligent labels for the description of the jewel collection available on the museum's Web site [27]. By using natural language generation techniques, ILEX, of which only a prototype was made, provided museum artifact descriptions that took into consideration both the level of knowledge of the user and the history of the interaction. Web pages for each object in the museum's collection were generated as the visitor navigated around the system's Web site. These were virtual descriptions in that they did not exist in any form prior to the user's request but were created on the fly by the system and adapted to the individual user's needs and behaviors. The result is called dynamic hypertext.

Given the wide range of different goals that visitors have, the benefits of dynamic hypertext are important for museums. The production of online texts is highly flexible and can be tailored to a level of detail beyond that achievable with static hypermedia. It is possible to introduce a significant amount of variation, presenting each user with object descriptions that are built on a personal level of knowledge, interests, and history of interaction. Adaptivity and dynamic hypertext are also the basis of another early online personalized application designed in 1998 by the National Research Center: the Carrara Marble Museum Web site's virtual guide, which accompanies visitors during the exploration of the online collection [49].

When the agent associated with the virtual guide is activated, the presentation of a work of art in the museum's collection is accompanied by an extra window dedicated to the comments of the virtual guide, which offers additional information to make the user's visit more interesting and pleasant. This extra information is personalized in the sense that it automatically takes into consideration the works of art, the artists, the materials, and the techniques previously seen by the user. The adaptive information that is dynamically generated refers to five different categories of content: introductory, summative, comparative, dissimilar, and peculiar. The priorities among information types are managed by a set of rules defined by the programmers (rule-based filtering) that take into consideration the profile chosen by the user when accessing the collection (tourist, student, or expert) and the items accessed.

A typical example of these techniques is provided by the Calendar application on New York's Metropolitan Museum of Art Web site. The principle is very simple: for all those who are bewildered by the amount of daily programming at the Met, the system offers the possibility of setting a personal profile indicating particular interests by selecting from the many event types and subjects on a specific form. Every time that visitor returns to the Web site to check the events on the daily calendar and clicks into My Met Calendar mode, through the use of cookies, only those events that fit the established criteria will be displayed for that particular date or range of dates.

By using the same personalization technique through online form registration, the Louvre museum is looking into a customized alert system for its Web site. Instead of providing everybody with standard information concerning the museum's activities, services, and products, this personalized alert system allows the virtual visitor to set an individual profile by defining personal interests on a fill-in form and providing an e-mail address. Whenever an exhibition, a conference, or a concert of interest is about to take place, the user may be automatically informed about it by e-mail or SMS text message.

The National Museum of Ethnology in Leiden, Netherlands, has developed an on-site application called "The tour of the world in 80 questions" that allows children aged 7–13 to print out a personalized tour plan of the museum based on a personal choice of subjects and continents [6]. The tour plan, which is colorful and easy for children to understand, includes a series of maps that help locate the objects, a brief description of the artifacts, and a list of related questions which the young visitors need to answer during their museum exploration.

References

1. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. SIGMOD **29**(2), 297–306 (2000)
2. Armentano, M., Amandi, A.: Goal recognition with variable-order markov models. In: IJCAI, pp. 1635–1640 (2009)
3. Balabanovic, M., Shoham, Y.: FAB: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
4. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: AAAI, pp. 714–720 (1998)
5. Boone, G.: Concept features in re:agent, an intelligent email agent. In: 2nd International Conference on Autonomous Agents (Agents '98), pp. 195–204 (1998)
6. Bowen, J.P., Filippini-Fantoni, S.: Personalization and the web from a museum perspective. In: Museums and the Web (2004)
7. Brajnik, G., Guida, G., Tasso, C.: User modeling in intelligent information retrieval. Inf. Process. Manag. **23**(4), 305–320 (1987)
8. Breese, J., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: 14th Conference on Uncertainty in Artificial Intelligence, UAI, pp. 43–52 (1998)
9. Broder, A.: A taxonomy of web search. SIGIR Forum **36**(2), 3–10 (2002)
10. Brusilovsky, P., Millán, E.: User models for adaptive hypermedia and adaptive educational systems. In: The Adaptive Web, pp. 3–53 (2007)
11. Bueno, D., David, A.: METIORE: A Personalized Information Retrieval System. Lecture Notes in Artificial Intelligence, vol. 2109, pp. 168–177. Springer, Heidelberg/New York (2001)
12. Burke, R.: The wasabi personal shopper: a case-based recommender system. In: 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence, pp. 844–849 (1999)
13. Burke, R., Hammond, K., Young, B.C.: Knowledge-based navigation of complex information spaces. In: 13th National Conference on Artificial Intelligence, pp. 462–468 (1996)
14. Charniak, E., Goldman, R.P.: Plan recognition in stories and in life. CoRR abs/1304.1497 (2013)
15. Chien, Y.H., George, E.: A bayesian model for collaborative filtering. In: Seventh International Workshop Artificial Intelligence and Statistics (1999)
16. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. **28**(4), 427–466 (2003)
17. Crook, P.A., Lemon, O.: Representing uncertainty about complex user goals in statistical dialogue systems. In: SIGDIAL Conference, pp. 209–212 (2010)
18. Deshpande, M., Karypis, G.: Selective markov models for predicting web page accesses. ACM Trans. Internet Technol. **4**(2), 163–184 (2004)

19. El-Sayed, M., Ruiz, C., Rundensteiner, E.: Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In: WIDM (2004)
20. Foltz, P., Dumais, S.: Personalized information delivery: an analysis of information filtering methods. Commun. ACM **35**(12), 51–60 (1992)
21. Fu, X., Budzik, J., Hammond, K.: Mining navigation history for recommendation. In: IUI (2000)
22. Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A.: User profiles for personalized information access. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. Lecture Notes in Computer Science, vol. 4321, pp. 54–89. Springer, Berlin/Heidelberg (2007)
23. Getoor, L., Sahami, M.: Using probabilistic relational models for collaborative filtering. In: WebKDD (1999)
24. Goldberg, D., Nichols, D., Oki, D., Terry, D.: Using collaborative filtering to weave an information tapestry communications of acm. Commun. ACM **35**(12), 61–70 (1992)
25. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: ACM SIGIR Conference, pp. 230–237 (1999)
26. Herrera, M.R., de Moura, E.S., Cristo, M., Silva, T.P.C., da Silva, A.S.: Exploring features for the automatic identification of user goals in web search. Inf. Process. Manag. **46**(2), 131–142 (2010)
27. Hitzeman, J., Mellish, C., Oberlander, J.: Generation of museum web pages: the intelligent labelling explorer. In: Museums and the Web (1997)
28. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. **22**(1), 89–105 (2004)
29. Jeh, G., Widom, J.: Scaling personalized web search. In: 12th International World Wide Web Conference (2003)
30. Joachims, T., Freitag, D., Mitchell, T.: Webwatcher: a tour guide for the world wide web. In: Proceedings of the International Joint Conference on Artificial Intelligence (1996)
31. Jokela, S., Turnpeinen, M., Kurki, T., Savia, E., Sulonen, R.: The role of structured content in a personalised news service. In: 34th Hawaii International Conference on System Sciences, pp. 1–10 (2001)
32. Kahraman, H.T., Sagiroglu, S., Colak, I.: The development of intuitive knowledge classifier and the modeling of domain dependent data. Knowl. Based Syst. **37**(3), 283–295 (2013)
33. Kautz, H., Selman, B., Shah, M.: Referralweb: combining social networks and collaborative filtering. Commun. ACM **40**(3), 63–65 (1997)
34. Kazienko, P.: Mining indirect association rules for web recommendation. Int. J. Appl. Math. Comput. Sci. **19**(1), 165–186 (2009)
35. Kent, A.: Knowledge-based recommender systems. In: Encyclopedia of Library and Information Systems, vol. 69, pp. 99–111. CRC Press, Boca Raton (2000)
36. Kießling, W., Kostler, W.: Foundations of preferences in database systems. In: VLDB Conference (2002)
37. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (1998)
38. Koutrika, G., Ioannidis, Y.: Personalization of queries in database systems. In: ICDE, pp. 597–608 (2004)
39. Koutrika, G., Ioannidis, Y.: Personalized queries under a generalized preference model. In: ICDE Conference (2005)
40. Krulwich, B., Burkey, C.: Learning user information interests through extraction of semantically significant phrases. In: AAAI Spring Symposium on Machine Learning in Information Access, Stanford (1996)
41. Lesh, N., Rich, C., Sidner, C.L.: Using plan recognition in human-computer collaboration. In: Proceedings of International Conference on User modeling, pp. 23–32. Springer, New York (1999)
42. Maragoudakis, M., Thanopoulos, A., Fakotakis, N.: Meteobayes: effective plan recognition in a weather dialogue system. IEEE Intell. Syst. **22**(1), 67–77 (2007). doi:10.1109/MIS.2007.14

43. Middleton, S., Shadbolt, N., Roure, D.D.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. **22**(1), 54–88 (2004)
44. Minio, M., Tasso, C.: User modeling for information filtering on internet services: exploiting an extended version of the umt shell. In: UM96 Workshop on User Modeling for Information Filtering on the WWW (1996)
45. Mooney, R., Roy, L.: Content-based book recommending using learning for text categorization. In: 5th ACM Conference on Digital Libraries, pp. 195–204 (2000)
46. Oberlander, J., O'Donell, M., Mellish, C., Knott, A.: Conversation in the museum: experiments in dynamic hypermedia with the intelligent labeling explorer. In: The New Review of Multimedia and Hypermedia, pp. 11–32 (1998)
47. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Tech. rep., Stanford University Database Group (1998)
48. Papadimitriou, D., Velegrakis, Y., Koutrika, G., Mylopoulos, J.: Goals in social media, information retrieval and intelligent agents. In: ICDE (2015)
49. Paterno, F., Mancini, C.: Designing web user interfaces adaptable to different types of use. In: Museums and the Web (1999)
50. Pazzani, M.J., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**(3), 313–331 (1997)
51. Pitkow, J., Schutze, H., Cass, T., Cooley, R., TurnBull, D., Edmonds, A., Adar, E., Breuel, T.: Personalized search. Commun. ACM **45**(9), 50–55 (2002)
52. Colace, F., Moscato, V., Quintarelli, E., Rabosio, E., Tanca, L.: Context awareness in pervasive information management. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Heidelberg/New York (2015)
53. Rafter, R., Bradley, K., Smyth, B.: In: Brusilovsky, P., Stock, O., Strapparava, C. (eds.) Adaptive Hypermedia and Adaptive Web-Based Systems. Lecture Notes in Computer Science, vol. 1892, pp. 363–368. Springer, Berlin/Heidelberg (2000)
54. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–58 (1997)
55. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Conference on Computer Supported Cooperative Work, pp. 75–186 (1994)
56. Riedl, M.O.: Narrative generation: balancing plot and character. Ph.D. thesis, North Carolina State University (2004)
57. Rose, D.E., Levinson, D.: Understanding user goals in web search. In: WWW, pp. 13–19. ACM, New York (2004)
58. Sarwat, M., Bao, J., Chow, C.Y., Levandoski, J., Magdy, A., Mokbel, M.F.: Context awareness in mobile systems. In: Data Management in Pervasive Systems. Springer, Heidelberg (2015)
59. Schafer, J.B., Konstan, J.A., Riedl, J.: Meta-recommendation systems: user-controlled integration of diverse recommendations. In: 11th International Conference on Information and Knowledge Management (CIKM) (2002)
60. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating word of mouth. In: ACM CHI Conference, pp. 210–217 (1995)
61. Sieg, A., Mobasher, B., Lytinen, S., Burke, R.: Concept based query enhancement in the ARCH search agent. In: International Conference on Internet Computing, pp. 613–619 (2003)
62. Sorensen, H., McElligot, M.: Psun: a profiling system for usenet news. In: CKIM 95 Workshop on Intelligent Information Agents (1995)
63. Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. ACM Trans. Database Syst. **36**(3), 19 (2011)
64. Tanudjaja, F., Mui, L.: Persona: a contextualized and personalized web search. In: 35th Hawaii International Conference on System Sciences (2002)
65. VanLehn, K.: Student models. In: Polson, M.C., Richardson, J.J. (eds.) Foundations of Intelligent Tutoring Systems. Laurence Erlbaum, Hillsdale (1988)
66. Yan, T., Garcia-Molina, H.: Sift: a tool for wide area information dissemination. In: USENIX Technical Conference, pp. 177–186 (1995)

Chapter 12
Context Awareness in Pervasive Information Management

Francesco Colace, Vincenzo Moscato, Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca

12.1 Introduction

Today we are living an epochal change, whereby the advent of the Internet and the development of communication technologies have completely transformed access to knowledge, whether it be structured data, services, Web documents, or others: as a result, in the plethora of available contents, finding the relevant knowledge is becoming more and more difficult. Concurrently, the advent of pervasive systems [26], where sensors and various other kinds of devices are seamlessly integrated into the user's world, has inspired plenty of research on the concept of *context*: context-aware applications and infrastructures adapt their behavior to the environmental conditions in which they are operating.

Context, as defined by Dey [11], is represented by the (meta)information that can be used to characterize the situation of any entity that is relevant for the system under consideration. For example, an intelligent room could trigger different actions based on the number of people present at each given moment, regulating the ventilation system in a different way.

F. Colace (✉)
Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata, Università degli Studi di Salerno, Via Ponte Don Melillo 1, 84084 Fisciano, Salerno, Italy
e-mail: fcolace@unisa.it

V. Moscato
Dipartimento di Ingegneria elettrica e delle Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy
e-mail: vmoscato@unina.it

E. Quintarelli • E. Rabosio • L. Tanca
Dipartimento di Elettronica, Informazione e Biongegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
e-mail: elisa.quintarelli@polimi.it; emanuele.rabosio@polimi.it; letizia.tanca@polimi.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_12

Surveys about context-aware systems can be found in [3, 5, 13, 15, 37, 40]. In the scope of knowledge access, the current context has often been adopted as a criterion for knowledge filtering [7]: the adaptation in this case includes providing the user only with the portion of knowledge that is relevant for the situation that she/he is experiencing, thus favoring easier access to what the user really needs.

The context can then be interpreted as the set of personal and environmental features determining the content to be delivered. Consider the museum scenario of our running example: useful context perspectives may be the kind of user (e.g., specialist, nonexpert, or museum technical operator), the interest topic (e.g., museums or artworks), the situation (e.g., alone, with friends, or with children), the time (e.g., morning, afternoon, or evening), and the location. For instance, the information provided to expert users who are visiting the museum alone will be different from that received by nonexperts visiting the museum with their children.

Focusing mainly on the applications of context to data and services, this chapter first introduces a *context model*, the context dimension tree (CDT) [8], which serves as a conceptual support for the design of context-aware systems, and then studies some interesting context-related issues.

The chapter is structured as follows: Sect. 12.2 provides a short introduction to the existing research on context-based knowledge access. Consistently with the general aim of the book, the rest of the chapter provides fundamental notions and methods that aim at guiding the reader toward the systematic design of context-aware systems. Section 12.3 presents the main components and activities characterizing every context-aware system and introduces the CDT model. Section 12.4 explains how contexts can be employed to deal with data [10] and services. Section 12.5 proposes a solution to the context schema evolution problem [23], arising when the set of possible contexts for a given application changes over time. Section 12.6 illustrates how context-aware data filtering may be further refined, exploiting the users' preferences in each given context [17, 18]. Section 12.7 describes the way to declare and enforce context within a pervasive data language, using the example of PerLa [28], while Sect. 12.8 draws some conclusions and proposes a few related research challenges.

12.2 Existing Approaches to Context-Based Knowledge Access

The approaches to context-based knowledge access may be classified on the basis of their aim: (1) using different perspectives (views) in data modeling, (2) partitioning information bases, (3) determining the set of relevant services, and (4) information filtering. Some relevant proposals for each category are listed in Table 12.1 and briefly described in the following text.

Table 12.1 Existing proposals on context-based knowledge access

Using different perspectives in data modeling	Stavrakas et al. [32–34], Roussos et al. [25], Martinenghi and Torlone [16], Guha et al. [12], and Stoermer et al. [36]
Partitioning information bases	Motschnig-Pitrik [20] and Theodorakis et al. [38]
Determining the set of relevant services	Raverdy et al. [24]
Information filtering	Stefanidis et al. [35], Bobillo et al. [6], Yang et al. [39], Adomavicius et al. [1], Shin et al. [31], and Baltrunas et al. [4]

12.2.1 Using Different Perspectives in Data Modeling

This category contains the approaches aimed at enriching existing data models to manage context-dependent data. In all these proposals, context is used to deal with ambiguities of data interpretation in different conditions, granting the possibility that the same entity assumes different values according to the context.

Stavrakas et al. [32–34] extend the syntax of XML in order to allow the specification of multidimensional documents, in which element content and attributes may vary according to the context; for instance, text and values can change on the basis of the adopted language and currency. The enriched XML syntax allows adding to document portions the contexts where they are valid; contexts can be specified providing one or more values for some dimensions. Contextual features are also integrated in the graph-based object exchange model (OEM), often used to deal with semistructured information. Moreover, a multidimensional query language named MQL is designed, and an implementation of MQL relying on Lorel, the existing query language for OEM, is proposed.

Similarly, the context has also been incorporated in the relational model by Roussos et al. [25]. In their framework, a tuple attribute may assume different values in different contexts; acting this way, a tuple has several variants, or *facets*, one for each context. The relational algebra operators are extended in order to take context into account; in particular, enhanced versions of projection, selection, Cartesian product, join, and set operations are provided. The work of Martinenghi and Torlone [16] proceeds analogously, defining a context-aware relational model and an extended relational algebra; in this case, the authors also prove interesting properties of the enhanced operators that can be useful for optimization purposes.

In the scope of RDF, Guha et al. [12] consider a problem, typical of the semantic Web, where data coming from diverse sources may contain contradictory statements, e.g., because they refer to different time instants or because of different interpretations of the meaning of certain classes. In the vision of Guha et al., each data source is associated with a context, representing the way it interprets data. The different contexts capture the differences among the data sources and are used when these need to be aggregated. To this aim, the authors propose rules to move data from one context to another. The issue of contradictory RDF knowledge bases is also covered by Stoermer et al. [36], who again treat the context as a way of interpreting data; in addition, the authors allow the specification of relations between contexts that are then used to support query answering.

12.2.2 Partitioning Information Bases

The strategies that follow are early attempts to employ context to deal with collections of entities of any kind and thus are not explicitly and exclusively focused on data.

Motschnig-Pitrik [20] uses context as a basis for decomposing generic information bases, where an information base can represent a database, an information system, object-oriented software, or others. An information base is composed of a set of information units, and the context is a subset of the information base. Contexts may be overlapping and are hierarchically organized through a containment relation. In different contexts, the same information unit can be assigned different names and different attributes. Operations called *transactions* are introduced to create and modify contexts, and policies to propagate the changes produced in a context to those it possibly contains are devised. This work also proposes a strategy to associate users with permissions to perform certain transactions.

The framework of [20] is extended by Theodorakis et al. [38], who especially study naming problems. They introduce the possibility for each information unit to have various names even in the same context and try to ensure that, after a transaction, each information unit in each context must be distinguished from the others by its name.

12.2.3 Determining the Set of Relevant Services

Raverdy et al. [24] present a context-aware middleware to manage service discovery and access. At any moment, the network, each user, and each service are endowed with a context, representing their current status and features. The context is used first of all to filter the wide amount of services retrieved to users in response to service discovery queries: the context of the user is compared with those of the services, and only the services with a context similar to that of the user are retained. Moreover, the context of the network, describing aspects like traffic and bandwidth, is used in combination with the users' contexts to optimize the routing of the service discovery queries.

12.2.4 Information Filtering

The last approaches we present employ the context to select relevant data items in large datasets.

Stefanidis et al. [35] propose a methodology to personalize the results of relational queries on the basis of *contextual preferences*, relying on a context model expressing contexts through a series of dimensions organized hierarchically. The main aim of the chapter is to study the *context resolution problem*, involving the determination of the set of preferences applicable to filter the result of a given query formulated in a certain context.

Bobillo et al. [6] describe an approach to context-aware data tailoring aimed at *improving a decision support system*. In more detail, the decision support system relies on a domain ontology that may be huge. Therefore, the authors propose tailoring ontology portions on the basis of the context, so that the user knowledge, in the various contexts, only consists of the appropriate portion. The context is modeled through an ontology too, providing also the possibility to specify hierarchies. A third ontology, containing all the classes envisaged by the previous ones, is used to establish the relationships between the contexts and the concepts of the domain ontology. The paper proposes an algorithm that, on the basis of the three ontologies, defines how to retrieve the suitable data instances for each given context.

Yang et al. [39] propose a context model for a *ubiquitous learning environment*, used to tailor the appropriate content to be given to each learner. The model is ontology based and allows to describe the context of both the user and the content; the appropriate content is thus discovered through rule-based matching between the two contexts. The proposed dimensions for the users are their personal features like preferences and expertise and their time and location; the context associated with the content is described by basic information like the title and then the publisher, the structure of the document, and annotations provided by past users. Techniques to acquire certain context dimensions through sensors like GPS and RFID are also proposed.

Context has been employed for information filtering also in the field of *recommender systems*. Adomavicius et al. [1] propose techniques to compute recommendations in a multidimensional scenario: context perspectives, like day and time, are added to the traditional dimensions representing users and items. The considered recommendation problem consists in suggesting interesting items on the basis of the knowledge about the user but also about his/her current situation. To this aim, a reduction-based approach is devised: users provide ratings to some of the items specifying a context, and only the ratings associated with contexts similar to C are taken into account to compute recommendations for the context C. Shin et al. [31], on the other hand, rely on a log of the past data accesses instead of using explicit ratings; however, similarly to [1], they compute the recommendations based on the similarity between the contexts in which the log data has been collected and the context in which the recommendations are required. Finally, Baltrunas et al. [4] discuss a context-aware recommender system for mobile devices, where recommendations are provided enhancing a conventional matrix-factorization model for collaborative filtering with contextual information. In addition, the authors describe a technique to choose the most appropriate context dimensions and values for a particular application.

Each of the approaches we have described recognizes the importance of context and tries to solve some specific and important problem; however, none of them shows any attempt to introduce a systematic way to represent, and deal with, the life cycle of a context-aware system. In the following text, we briefly illustrate a general framework where different context-related issues can find a solution within a unified design methodology. This methodology constitutes the basis of many research projects, including Context-ADDICT [9], PerLa [28–30], ADaPT [19], and SuNDroPS [21], and has already been applied to the design of the context-aware components of the Green Move [22], Motus,[1] and Databenc[2] systems.

12.3 Context-Aware Knowledge Adaptation

Independent of the specific application scenario, the systematic design and management of a context-aware system have to contemplate a common set of steps, both at design time and at runtime, leading from the definition of the possible contexts to the realization of the actual context-aware behavior. To be more precise:

- At **design time** (see Fig. 12.1), using a *context model*, the designer must conceive a *context schema* for the considered scenario, that is, a representation of the available contexts describing the relevant perspectives with respect to which the context may change—the so-called dimensions—and the values they can take. These values may correspond to data that, at runtime, can be obtained from physical sensors (e.g., the present location) or by interacting with other applications (e.g., to obtain the date from an electronic calendar) or by directly asking the user (e.g., to know his/her current mood). The contexts represented in the context schema must then be associated with the related context-aware knowledge, which naturally derives from the application scenario and normally consists of pieces of information (represented in terms of *views* over the available data), useful services, rules to be triggered, etc.
- At **runtime** (see Fig. 12.2), the system recognizes the current context by gathering the necessary information from applications and physical sensors, or asking the user when necessary, and triggers the related context-aware behavior defined at design time. Note that when context information comes from physical sensors or other applications, the obtained raw data usually needs to be preprocessed in order to be associated with one of the values of the corresponding dimensions.

[1] http://www.motus2015.it/en

[2] http://www.databenc.it

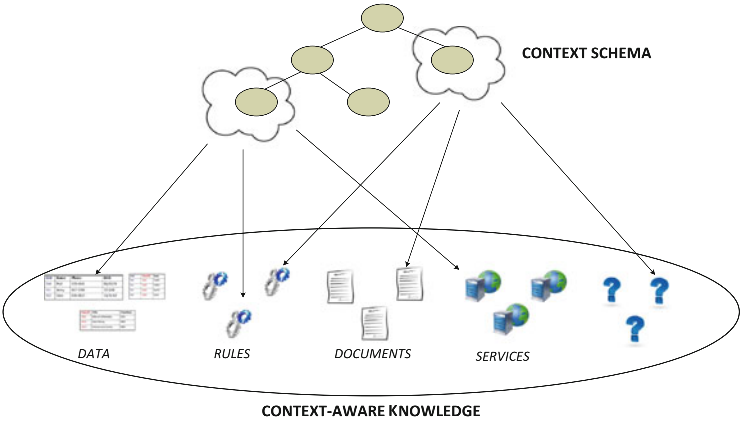


Fig. 12.1 Operations required during the design of a context-aware system: a context schema is designed, and the contexts that it represents are associated with the appropriate knowledge

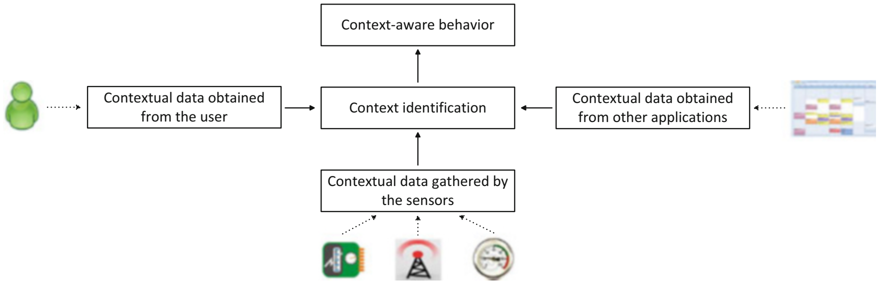

Context-aware behavior
Contextual data obtained from the user
Context identification
Contextual data obtained from other applications
Contextual data gathered by the sensors

Fig. 12.2 Operations executed at runtime: the context is identified and used to select the behavior to be triggered

For instance, a numerical temperature value would need to be recognized as belonging to the appropriate class (e.g., cold, warm, or hot), or the set of Web pages the user is viewing should be associated with his/her current interest topic.

Context has often been regarded with the exclusive aim of dealing with mobility issues—see also [27]. In the following text, we describe the CDT model that, on the contrary, permits to uniformly manage context information of various types, possibly coming from diverse sources.

The CDT model allows to represent context schemas as trees with nodes of two kinds: *dimensions* and *dimension values*, also known as "concept nodes." Figure 12.3 shows a possible context schema in the museum domain of the running example. Dimensions are represented as black nodes, while their values are drawn as white nodes. The dimension nodes represent the different perspectives describing a context, e.g., the type of `user` and the `situation`; examples of possible values for the `user` dimension from the schema of Fig. 12.3 are `specialist`

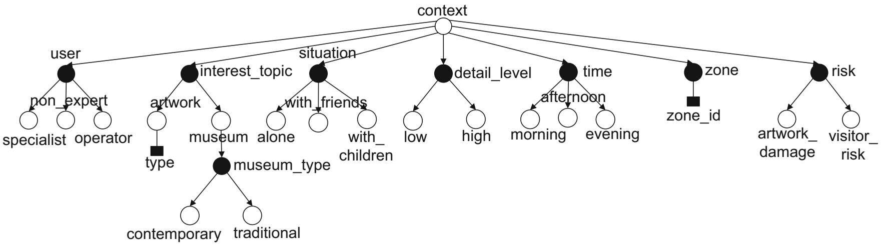


Fig. 12.3 A context schema

and non_expert. The root is a special value node representing the most general context, and its children are the main analysis dimensions. Every dimension has only value children, and each value has only dimension children; the latter are subdimensions further specifying the value. Dimensions and values can be endowed with attributes that are parameters whose values can be dynamically derived from the environment or provided by the users themselves at execution time. A dimension can be connected to at the most one attribute, used to replace a huge number of values when it is impractical to enumerate them all (e.g., a GPS location). If this is the case, the dimension does not have any value child, e.g., the zone dimension has as child the attribute zone_id. One or more attributes can be added to values too: such attributes are used to select specific instances from the set of values represented by a value node (e.g., the artworks of a certain type). Attributes are graphically represented in Fig. 12.3 by square nodes.

A context defined according to a context schema is a conjunction of (any number of) dimension-value equalities. To each dimension is assigned one of its value children, possibly specifying attribute values in the form *$attribute_name=attribute_value* or just a pair *$attribute_name=attribute_value*; the latter form is reserved to those dimensions carrying an attribute instead of value children. The conjuncts composing the contexts are called *context elements*. For instance, the context in Listing 12.1 represents a nonexpert user, who is located in the zone identified as 80131, is visiting museums during the evening with friends, and is interested in paintings.

Listing 12.1 Example of a context representing a nonexpert user

```
user=non_expert ∧ interest_topic=artwork(type=``painting") ∧←↩
     situation=with_friends ∧ time=evening ∧ zone=($zone_id←↩
    =``80131")
```

Note that this context does not specify any values for the detail_level and risk dimensions. Indeed, it is not required that all the dimensions have a value.

12.4 Using Context to Deal with Data and Services

12.4.1 Context-Aware Definition of Relevant Areas in Databases

When applying context to data management to perform information filtering, the context-aware behavior is represented by the selection—or *tailoring*—of a part of a dataset to be delivered to the user; thus, in the design phase, the possible contexts must be associated with *portions of this dataset*. In order to convey the basic idea, and without loss of generality, we consider data as stored in a unique database. In general, do note that the data in the database may be the result of the integration of diverse sources, like relational stores or semistructured repositories; part of the information might even be gathered from the environment, e.g., through the PerLa middleware [28]. This hypothesis does not impair the generality of the methodology, since the only assumption we make is the existence of a (possibly loose) global data representation such as a relational schema or an ontology. In the examples below, we use the following (subset of a) relational database schema in the museum domain, featuring information about museums, artworks, and artists:

ARTWORK(artwork_id, name, type, museum_id, artist_id, description)
MUSEUM(museum_id, zone_id, museum_type, opening_time, closing_time)
ARTIST(artist_id, birth_year, biography)

Each context C envisaged by the context schema must be associated with a subset (a *view*) of the whole database, called *relevant area*, and denoted by $\mathscr{R}el(C)$. For instance, for relational databases, each relevant area is formed by some database tables, each restricted, in turn, to a subset of its attributes and tuples. However, note that the number of possible contexts may be huge even for small context schemas; thus, the naive approach of manually specifying a relevant area for each context turns out to be unfeasible.

Since the relevant area of a certain context should contain the information that is pertinent to all the elements composing it, this relevant area should be computed as the *intersection* of the relevant areas of the component elements. Thus, a possible solution to support the designers is requiring them to explicitly define relevant areas *only for the contexts constituted by just one element* and then automatically derive the relevant areas for the complex contexts as their intersection. For instance, $\mathscr{R}el(\texttt{time} = \texttt{evening} \wedge \texttt{detail\_level} = \texttt{low})$ should contain the data related to `evening` as well as to `low`.

The technique to be used in the computation of the intersection depends on the adopted data model; for example, for relational databases, we use the *double intersection* ($\Cap$) operator, whose operands are *sets of relational (virtual) views* instead of single relations [10]. Let us analyze how double intersection works on our relational running example. The relevant area of $\texttt{time} = \texttt{evening}$ contains, for all the tables of the database, only the information about the museums open in the evenings and the artworks exhibited in these museums:

$$\mathscr{R}el(\texttt{time} = \texttt{evening}) = \{\ \sigma_{closing\_time>20}\textsc{Museum},\ \textsc{Artwork} \ltimes \sigma_{closing\_time>20}\textsc{Museum},\ \textsc{Artist} \ltimes \textsc{Artwork} \ltimes \sigma_{closing\_time>20}\textsc{Museum}\}$$

The context $\texttt{detail\_level} = \texttt{low}$ instead does not apply any filter on tuples but excludes some relations and attributes:

$$\mathscr{R}el(\texttt{detail\_level} = \texttt{low}) = \{\ \pi_{artwork\_id,name,type,museum\_id}\textsc{Artwork},\ \textsc{Museum}\}$$

As a consequence, the relevant area of the complex context $\texttt{time} = \texttt{evening} \wedge \texttt{detail\_level} = \texttt{low}$ must include the data that is pertinent to both the component context elements:

$$\mathscr{R}el(\texttt{time} = \texttt{evening} \wedge \texttt{detail\_level} = \texttt{low}) = \mathscr{R}el(\texttt{time} = \texttt{evening}) \Cap \mathscr{R}el(\texttt{detail\_level} = \texttt{low}) = \{\ \pi_{artwork\_id,name,type,museum\_id}\textsc{Artwork} \ltimes \sigma_{closing\_time>20}\textsc{Museum},\ \sigma_{closing\_time>20}\textsc{Museum}\ \}$$

Note that the double intersection retains the tables and attributes that are present in *all* the relevant areas of the involved context elements and then performs the intersection of their contents. In general, it is clear that the number of possible context elements is much lower than the total number of contexts; therefore, in general, this method reduces significantly the designer effort.

The designer effort may be further alleviated by progressively computing the relevant areas associated with the internal nodes as the union of those associated with their direct descendants, starting from the leaves. In this way, the designer has to provide relevant areas only for the latter. In a similar way as above, in the relational case we use the operator *double union* ($\Cup$) which builds the union among sets of relational views. For instance, the relevant area of $\texttt{interest\_topic} = \texttt{museum}$, instead of being manually defined, can be computed as the double union of the relevant areas of its leaf descendants:

$$\mathscr{R}el(\texttt{interest\_topic} = \texttt{museum}) = \mathscr{R}el(\texttt{museum\_type} = \texttt{contemporary}) \Cup \mathscr{R}el(\texttt{museum\_type} = \texttt{traditional})$$

The area derived through the double union features all the tables and attributes contained in at least one of the involved context elements. For the tables that are relevant for both `contemporary` and `traditional`, the union of the tuples is computed.

As may be expected, the automatic procedure based on the double intersection may be less precise than a completely manual definition of the relevant areas: a final revision is thus advisable to introduce the necessary refinements.

Note that the method summarized above puts all the burden of defining contexts and context-aware data on the designer, also supposing that a designer is able to imagine which data are relevant in each given context. In a more dynamic and self-adaptive scenario, the system should be able to infer the relevant area expressions from the past activities of the users through some inductive techniques, automatically learning from the users' behaviors (e.g., their queries in the different contexts) what are the relevant data of each context. The development of such a methodology is part of our ongoing work. Since the definition of the relevant areas in this case is automatic, the strategy we are studying associates a suitable data portion with each possible context, without the need to resort to double intersection for combining the relevant areas of the context elements.

12.4.2 Services' Activation Driven by Context

For similar purposes to those seen for *data*, here we investigate the possibility of tailoring only a subset of *services*[3] really necessary for users—equipped with their personal devices—depending on the context. Thus, the context awareness is integrated with—yet orthogonal to—data and service management, where the knowledge of the context in which the data and services are used drives the process of focusing on currently useful information (represented by means of relevant database areas) and applications.

As a simple motivating example, we consider tourists that during their vacation in Campania desire to visit a special exhibition of pictures by Michelangelo Buonarroti in the *National Museum of Capodimonte* in Naples.

We suppose that the museum provided a set of *smart* services, as an example, for booking a visit for a specific date, buying the related ticket, user accounting and registration, accessing proper multimedia guides describing the artworks' story, recommending special visit paths, monitoring the environmental condition of each room, showing the exit in the case of emergency situations, saving the visit in a digital format, etc.

The major problems are related to the fact that not all the discussed services are effectively useful and similarly not all the related data are really of interest for a given user. It depends on the particular contexts observable for a given user interacting with the described environment; for example, the multimedia guide service and information about Michelangelo's pictures are not useful for tourists who have not yet booked the museum visit.

To For these reasons, a context-aware system able to tailor useful data and service for users depending on the context should be desirable.

Once we have modeled the context by means of the CDT approach, we then represent the possible temporal evolution of context instances that in our hypothesis is driven by the *events* that correspond with particular *conditions* (e.g., a sensor detects a very high temperature value for a museum room) and software *services* (e.g., booking a museum visit or using a multimedia guide).

[3] The term service is used to indicate a set of software functionalities that can be reused for different aims.

Indeed, the conditions' occurrence or services' activation may determine a *context switch* and at the same time the dynamic selection of "useful" data (*contextual relevant areas*) and services (*basket of services*) that can be opportunely exploited in the new context as a result of data and service tailoring.

Several models and formalisms can be exploited for our aims. We decide to adopt a Petri net structure, named *context evolution Petri net* (CEPN), able to model the evolution among context instances depending on the conditions or services that follow one another in the considered environments for a given user. In particular, we propose a simplified and non-probabilistic model of the constrained Probabilistic Petri Net (PPN) that some of the authors have applied for human activity recognition in video surveillance applications [2].

Given a set $C = \{c_1, \dots, c_n\}$ of contexts, a set $E = \{e_1, \dots, e_m\}$ of events, a set $\mathscr{S} = \{S_1, \dots, S_k\}$ of baskets of services, and a set $V = \{v_1, \dots, v_h\}$ of contextual relevant areas, we can model for each user role in a given scenario the context evolution as a tuple $CEPN = \{P, T, \longrightarrow, l_p, l_t\}$, satisfying the following properties:

1. P and T are finite disjoint sets of *places* and *transitions*, respectively, i.e., $P \cap T = \emptyset$.
2. $\longrightarrow$ is the flow relation between places and transitions, i.e., $\longrightarrow \subseteq (P \times T) \cup (T \times P)$.
3. $l_c : p \in P \rightarrow (c, v, S) | c \in C, v \in V, S \in \mathscr{S}$ is a *labeling function* that associates each place with a context label and the related contextual relevant area and basket of services.
4. $l_s : (t, e) \rightarrow 2 | t \in T, e \in E$ is an *enabling function* that associates each transition with the event that can enable the transition, i.e., a particular service (or a services' composition) that can be activated or condition that can occur in the current state originating the context switch. In the case of events corresponding to services' activation, there is a constraint that the activated services have to belong to the basket of services related to the current place.
5. The *preset* of a node $x \in P \cup T$ is the set $\{y | y \rightarrow x\}$. This set is denoted by the symbol $\hat{x}$. $\exists x \in P$ s.t. $\check{x} = \emptyset$. This means that there exists at least one *terminal node* in the Petri net.
6. The *postset* of a node $x \in P \cup T$ is the set $\{y | x \rightarrow y\}$. This set is denoted by the symbol $\check{x}$ $\exists x \in P$ s.t. $\hat{x} = \emptyset$. This means that there exists at least one *start node* in the Petri net.

The dynamics of a Petri net are represented via "markings." For a $CEPN = \{P, T, \longrightarrow, l_p, l_t\}$, a *marking* is a function $\mu : P \rightarrow N$ that assigns a number of tokens to each place in P.

The execution of a Petri net is controlled by its current marking. A transition is enabled if and only if all its input places (the preset) have a token. When a transition is enabled, it may *fire*. When a transition fires, all enabling tokens are removed from the input place, and a token is placed in each of the output places of the transition (the postset): as transitions fire, the marking changes. A terminal marking is reached when one of the terminal nodes contains at least one token.

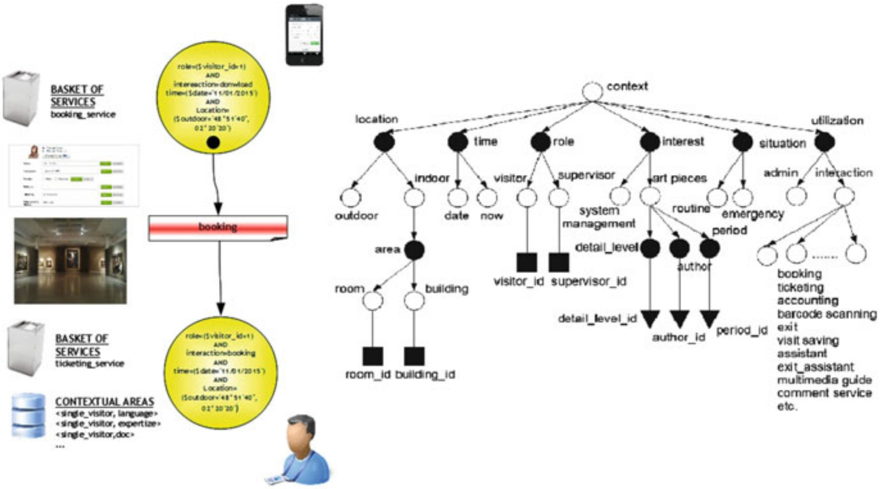


Fig. 12.4 Example of a part of CEPN with the related CDT

In the simplest case, all the enabled transitions may fire. However, to model more complex scenarios, we can impose other constraints to be satisfied before an enabled transition can fire. This set of constraints is represented by l_s. In other words, if a user is recognized in a given context (a token is present in the related CEPN place) and a particular event occurs, then the transition can fire and the user is recognized as belonging to different contexts (postset). As a consequence, a CEPN instance is able to drive the context evolution for a given user generating a single token in the start node when the related context instance has been recognized. Only while a terminal node has been reached, the token is removed from the CEPN and the related context evolutions can be reconsidered for the same user (Fig. 12.4). Figure 13.1 describes a situation represented by a CEPN with the related CDT in which a generic visitor desires to book a visit to a permanent exhibition at a museum by means of a dedicated mobile app. Initially, a user can only download from a dedicated Website the mobile application that allows the exhibition booking. Once the Web server has registered in its log information about user download, the related context instance is generated, and the system recognizes that the new context associated with the user corresponds to the state defined in the CEPN start node. In this state, no useful contextual data are associated with the user, while only the museum *booking* Web service is available. In particular, the Web service allows the booking of a visit for a particular date and requires some information about the user profile. Once a visit for a given date has been booked, the transition can fire and the next context contains information about the user profile in terms of favorite language, documentation type (images, video, audio) that should be provided by the museum multimedia guide app, and level of expertise; eventually, the *ticketing* service is enabled.

After the activation of the ticketing service, the user can buy the ticket and download all the museum apps.

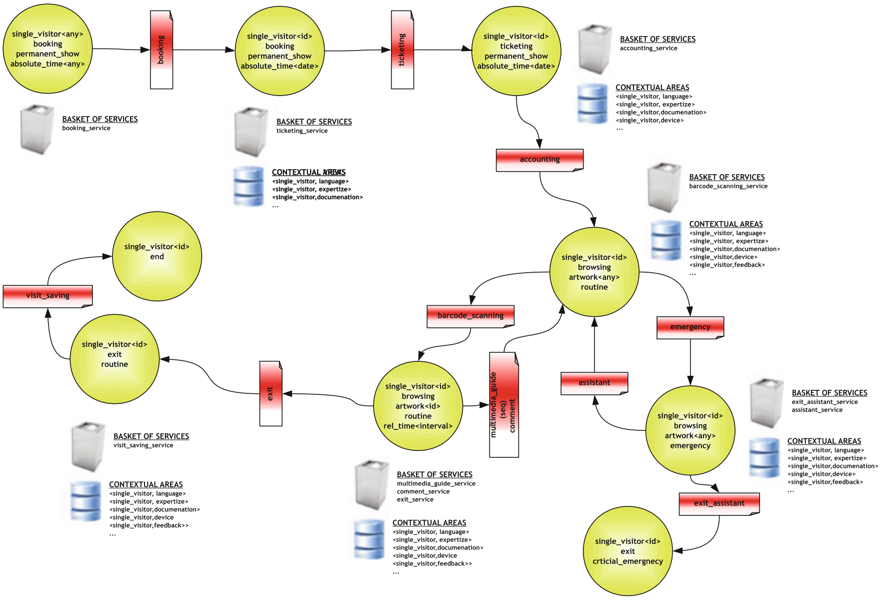


Fig. 12.5 A part of CEPN for a visitor user

Once the right context for a user is recognized, the evolution from one context instance to the other is driven by events: satisfaction of conditions inferred by sensors or activation of Web services.

Figure 12.5 schematizes the entire CEPN related to a *visitor* for the indoor museum context.[4] On the visiting date, users can activate the *accounting* service that registers their presence inside the museum. At this point, users can start their visit and only a *barcode scanning* service is available.

The *multimedia guide* service will be activated if and only if the user device reads a barcode related to a picture tailoring her/his preferences; possibly, a user can post on different social networks some feedbacks about the related experience using the *comment* service. By means of multimedia guide service, the user can be driven through the show and watch all the pictures of interest. At the end of the visit, users can terminate their experience activating the *exit* service and can choose to save her/his cultural path (observed pictures and the related documentation) in digital format using the *visit saving* service.

If during a visit an emergency event occurs, users have to follow the instructions provided by the *assistant* services (e.g., to reach the nearest exit or to change their path).

[4] For the sake of simplicity, we represent a context as a sequence of attributes/values in the form $dimension < value >$.

The definition in the design phase of a CEPN together with the related CDT for a given smart environment allows to capture the evolution of context instances depending on the sequence of events and to tailor in an effective manner data and services really useful to final users in relation to several situations that can occur.

12.5 Context Schema Evolution

The dimensions useful to describe the context, as well as their possible values, may change over time because of various reasons, including evolving business needs and technology developments. When the context schema changes, it may happen that some context-aware applications keep on using obsolete versions of the schema, thus requesting from the system knowledge related to contexts that are no more valid. Since these applications should still be able to obtain the appropriate context-aware behaviors, the schema evolution manager supports the transition by making the old contexts still "usable."

In the information filtering perspective, this implies also revising the associations between contexts and relevant areas according to the context schema modification.

Context schema evolution is managed through a sound and complete set of operators that the designer must use when he/she wants to modify the schema. Each operator modifies the context schemas in a precise way and is associated with the changes to be applied to the old context schemas to make them compliant with the new one. When the information filtering task is considered, each operator is also associated with the nodes of the context schema whose relevant areas need revision. If the designer uses these operators to define evolutions, the system can control it and react to schema changes in such a way that the applications are not ill-affected by the change.

We explain how context schema evolution works on an example where the aim is data filtering. Let us consider the context schema in Fig. 12.3 and suppose that at a certain point the application requirements change, and it is no more convenient to distinguish between people visiting museums alone or with friends: the only difference that is still considered important is that involving the presence of children. The designer, therefore, decides to merge the two nodes `alone` and `with_friends` in a unique node `without_children`. To perform this schema change, the designer applies the *Merge* operator, obtaining the updated schema in Fig. 12.6. After the change, the system immediately notifies the designer that a relevant area for the new context element $\texttt{situation} = \texttt{without\_children}$ must be provided.

Moreover, suppose that after the schema change, a client application still relying on the old schema requests the data portion associated with the context $\texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{evening}$. However, since the context-data associations are kept updated only for the current context schema, it is no more possible to retrieve the relevant area of the context indicated by the client. Therefore, such a context is updated on the basis of what is specified by the *Merge*

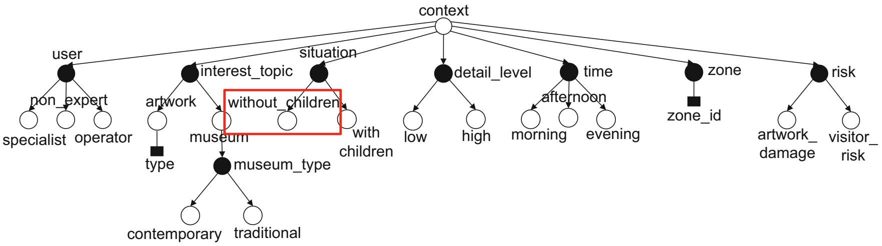


Fig. 12.6 Context schema of Fig. 12.3 after the application of the *Merge* operator

operator, making it compliant with the new schema. The operators define update functions for the contexts in such a way to preserve as much information as possible. In this case, the context $\texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{evening}$ is converted into $\texttt{situation} = \texttt{without\_children} \wedge \texttt{time} = \texttt{evening}$, and the relevant area of this context is delivered to the client application.

12.6 Data Personalization Based on Contextual Preferences

Context-based data tailoring assigns the same data to all the users sharing the same context. However, different users, even in the same context, may show different tastes. The relevant area defined at design time for a certain context, therefore, may be further filtered for each specific user, on the basis of his/her *preferences* in that context. In this section we discuss the usage of contextual preferences for personalizing context-aware relevant areas; see [14] for a general survey on data personalization.

The preference-based refinement of contextual data has been studied in [17, 18] in the scope of relational databases, relying on a preference model envisaging two kinds of preferences: preferences on tuples, or σ*-preferences*, and preferences on attributes, or π*-preferences*. σ-preferences are triples of the form $(C, SQ, score)$, where C is a context, SQ is a relational algebra expression identifying the tuples involved in the preference, and *score* is a value representing the preference score. Similarly, π-preferences are triples $(C, A, score)$, where C is a context, A is the attribute involved in the preference, and *score* is the preference score. Preference scores are real values in the range [0, 1], where values greater than 0.5 represent positive preferences, values lower than 0.5 indicate negative preferences, and 0.5 states indifference.

Miele et al. in [17] have proposed a methodology to apply contextual preferences to filter context-aware relevant areas, while paper [18] has studied how to automatically discover the preferences from logs of the past user activities. These issues will be briefly analyzed in Sects. 12.6.1 and 12.6.2.

12.6.1 *Using Contextual Preferences to Filter Context-Aware Data*

The personalization methodology described in [17] starts with the relevant area of a given context C and exploits the user's σ- and π-preferences to filter tuples and attributes. The personalization is composed of the following four phases:

1. **Preference selection**, selecting the subset of the user's preferences that are relevant for context C
2. **Attribute ranking**, exploiting the relevant π-preferences to assign a score to all the attributes
3. **Tuple ranking**, exploiting the relevant σ-preferences to assign a score to all the tuples
4. **Relevant area personalization**, selecting an appropriate subset of attributes and tuples on the basis of the computed scores

We explain the different phases with an example. Consider the context $C = \texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{evening}$, and suppose that the view on the ARTWORK table contained in the relevant area of C is that shown in Fig. 12.7. Moreover, suppose that the user has expressed the three preferences in Fig. 12.8.

The *preference selection* phase extracts from the preference set the preferences that are relevant for C, i.e., those associated with a context equal to or more general than C. In the example, σ_1 and π_1 are relevant, while σ_2 is not relevant. The context of σ_1, in fact, is $\texttt{time} = \texttt{evening}$ that is clearly more general than C, while π_1 is associated exactly with C. σ_2, on the contrary, specifies the context element $\texttt{time} = \texttt{daytime}$ that is not contained in C.

The relevant preference π_1 is used in the *attribute ranking* phase to compute a score for the attributes. The score of the preference (0.9) is assigned to the specified attribute *title*. All the other attributes are decorated with the indifference score 0.5, with the exception of the primary key *artwork_id* that receives the highest assigned score (0.9); this prevents discarding primary key attributes. In case more conflicting

Fig. 12.7 Sample ARTWORK table

ARTWORK			
artwork_id	**name**	**type**	**artist_id**
a1	Artwork1	painting	Artist1
a2	Artwork2	sculpture	Artist1
a3	Artwork3	painting	Artist3

Fig. 12.8 Sample preferences

$$\begin{aligned}
\sigma_1 = {} & (\texttt{time} = \texttt{evening}, \mathit{type} = \mathit{painting}, 0.8)\\
\sigma_2 = {} & (\texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{morning},\\
& \mathit{type} = \mathit{sculpture}, 0.75)\\
\pi_1 = {} & (\texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{evening},\\
& \mathit{name}, 0.9)
\end{aligned}$$

preferences were present for the same attribute, a formula to compute a combined score is defined in [17].

Similarly, the preference σ_1 is employed in the *tuple ranking* phase, assigning its score (0.8) to the paintings $a1$ and $a3$. The remaining artwork $a2$ obtains the indifference score. Again, techniques to manage the cases where conflicting preferences are present are provided in [17].

The last phase, the *relevant area personalization*, starts considering the attributes, removing those with a score lower than a specified threshold. In our example, supposing a threshold equal to 0.7, only *artwork_id* and *name* would be retained. Then, tuples are ordered on the basis of their scores, and the highest ranked ones are included in the relevant area until memory is available. In the example, artworks $a1$ and $a3$ are ranked first; the number of tuples actually included in the filtered relevant area, however, depends on the amount of memory available on the client device.

12.6.2 Mining Contextual Preferences

Requiring the user to manually specify his/her preferences in all the possible contexts, whose number may be huge, is unfeasible. This is why we have studied a strategy to automatically infer contextual preferences from past users' activities [18]. The phases leading to the discovery of the preferences are summarized in Fig. 12.9. First, the SQL queries performed by the user application are logged on the client device in the so-called device log, which is then incorporated into a global repository server-side called *server log* (phase 1). The server log is used to mine *association rules* (phase 2) that are finally converted into σ- and π-preferences by computing suitable scores in the range [0, 1] (phase 3).

Let us analyze the mining process with an example. Consider a device log containing only the following query: *SELECT name FROM artwork WHERE type= 'painting'*. This query is issued against the ARTWORK table in Fig. 12.7, in the context $\texttt{situation} = \texttt{with\_friends} \wedge \texttt{time} = \texttt{evening}$.

We start with σ-preferences. The server log is composed of a table for each relation of the database; each server log table contains the results of the queries performed by the user on the corresponding database table. Therefore, the artworks $a1$ and $a3$, results of the above query, are imported into the server log of ARTWORK, as shown in Fig. 12.10. Note that the server log records also the context in which the data has been accessed. The server log is then used to mine association rules of the form *context* → *conditions on data*. In the server log of Fig. 12.10, for instance,

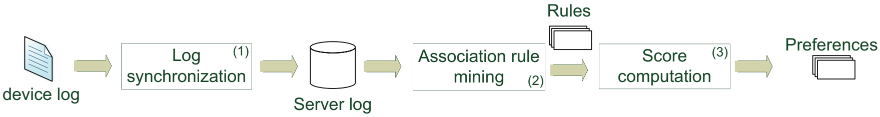


Fig. 12.9 Phases of the contextual preference mining

situation	time	...	artwork_id	name	type	artist_id	...
with_friends	evening		a1	Artwork1	painting	Artist1	
with_friends	evening		a3	Artwork3	painting	Artist3	

Fig. 12.10 Server log for the computation of σ-preferences

situation	time	...	artwork_id	name	type	artist_id	...
with_friends	evening			1	1		

Fig. 12.11 Server log for the computation of π-preferences

we can mine the rule *situation=with_friends* → *type='painting'*, with confidence 1. After the rule has been extracted, the system computes the frequency of the paintings in the dataset accessed by the user (here 0.67) and introduces a preference for paintings in the context `situation` = `with_friends`, with a score derived on the basis of this frequency and the rule confidence.

The procedure to mine π-preferences starts from a different server log, containing just a row for each query in the device log, indicating with "1" the columns mentioned in the *select* and *where* clauses of the query (see Fig. 12.11). On such a log, association rules of the form *context* → *attribute* are mined, and a preference score is derived on the basis of the rule confidence.

12.7 PerLa for Context

Adding context awareness to a pervasive system requires a clear definition of context and of the means to extract context parameter values from the real world: the first task is accomplished in the previous sections, while for the second task, we can resort to PerLa [28]. On the other hand, if the network, or more correctly the language handling it, is able to manage the context the sensors are related with, the relationship between context and data is correctly mastered in two directions. Indeed, while the sensors contribute by gathering the context values, they themselves can be influenced by their context and react accordingly. PerLa has been extended to manage the sensors' contexts explicitly, with a full declarative approach based on the CDT, allowing to create and maintain the context [30]. The syntax of the context manager is described in Listing 12.2.

Listing 12.2 Example of the context manager syntax

```
CREATE CONTEXT <ContextName>
ACTIVE IF <DimList>
[ SET [ <Enabled> ] [ <ActivateOn> ] [ <DeactivateOn> | <↩
    LifeTime>]]
[ ON ENABLE '{' <Query> '}' ]
[ ON DISABLE '{' <Query> '}' ]
REFRESH  EVERY <Duration>
```

Intuitively, $<$ `DimList` $>$ is a list of dimensions associated with their values, defining a context; the `SET` keyword is used to establish in which conditions this context is active; the `ON ENABLE` clause specifies the behavior associated with the context; while the `ON DISABLE` clause specifies the actions to be taken when the context is deactivated. The `REFRESH EVERY` clause specifies the rate at which these conditions have to be checked.

Let us consider our sample context schema in Fig. 12.3. The value `artwork_damage` of the `risk` dimension is activated when the sensors measure humidity and temperature values which can be critical for the preservation of artworks. The context manager expression in Listing 12.3 creates a context named *HumidityTemperatureMonitoring*, associated with the `user` = `operator` and `risk` = `artwork_damage` dimension-value pairs. When this context occurs, since the artworks could be subject to possible injuries, the operator starts receiving updates about humidity and temperature data for monitoring.

Listing 12.3 Example of context manager expression

```
CREATE CONTEXT HumidityTemperatureMonitoring
ACTIVE IF user=operator AND risk=artwork_damage
ON ENABLE (HumidityTemperatureMonitoring)
    SELECT humidity, temperature, device_location
    SAMPLING EVERY 1m
ON DISABLE
    DROP HumidityTemperatureMonitoring
REFRESH EVERY 10m
\end{verbatim}
```

12.8 Conclusion

This chapter has presented an overview of some issues related to context-aware knowledge access. The description has been founded on an existing context model, named CDT. Particular attention has been devoted to the system life cycle and to the problems connected to the efficient association between contexts and knowledge chunks. Moreover, a technique to deal with context schema evolution has been illustrated. Also, contextual preferences have been considered, explaining how they can be employed to refine context-aware data and how they can be automatically discovered from the past user activities. Finally, the integration of context awareness in the middleware of a pervasive system has been discussed.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)
2. Albanese, M., Chellappa, R., Cuntoor, N.P., Moscato, V., Picariello, A., Subrahmanian, V.S., Udrea, O.: A constrained probabilistic Petri net framework for human activity detection in video. IEEE Trans. Multimedia **10**(6), 982–996 (2008). doi:10.1109/TMM.2008.2001369. http://dx.doi.org/10.1109/TMM.2008.2001369
3. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
4. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Pers. Ubiquit. Comput. **16**(5), 507–526 (2012)
5. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
6. Bobillo, F., Delgado, M., Gómez-Romero, J.: Representation of context-dependant knowledge in ontologies: a model and an application. Expert Syst. Appl. **35**(4), 1899–1908 (2008)
7. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. **36**(4), 19–26 (2007)
8. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: Context information for knowledge reshaping. Int. J. Web Eng. Technol. **5**(1), 88–103 (2009)
9. Bolchini, C., Orsi, G., Quintarelli, E., Schreiber, F.A., Tanca, L.: Context modeling and context awareness: steps forward in the Context-ADDICT project. IEEE Data Eng. Bull. **34**(2), 47–54 (2011)
10. Bolchini, C., Quintarelli, E., Tanca, L.: CARVE: context-aware automatic view definition over relational databases. Inf. Syst. **38**(1), 45–67 (2013)
11. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
12. Guha, R.V., McCool, R., Fikes, R.: Contexts for the semantic web. In: Proceedings of ISWC 2004, 3rd International Semantic Web Conference, pp. 32–46. Springer, Heidelberg (2004)
13. Hong, J., Suh, E., Kim, S.: Context-aware systems: a literature review and classification. Expert Syst. Appl. **36**(4), 8509–8522 (2009)
14. Koutrika, G.: Data personalization. In: Francesco, C., et al. (eds.) Data Management in Pervasive Systems. Springer, Heidelberg (2015)
15. Lee, S., Chang, J., Lee, S.-g.: Survey and trend analysis of context-aware systems. Inf. Int. Interdiscip. J. **14**(2), 527–548 (2011)
16. Martinenghi, D., Torlone, R.: Querying context-aware databases. In: Proceedings of FQAS 2009, 8th International Conference on Flexible Query Answering Systems, pp. 76–87. Springer, Heidelberg (2009)
17. Miele, A., Quintarelli, E., Tanca, L.: A methodology for preference-based personalization of contextual data. In: Proceedings of EDBT 2009, 12th International Conference on Extending Database Technology, pp. 287–298. ACM, New York (2009)
18. Miele, A., Quintarelli, E., Rabosio, E., Tanca, L.: A data-mining approach to preference-based data ranking founded on contextual information. Inf. Syst. **38**(4), 524–544 (2013)
19. Miele, A., Quintarelli, E., Rabosio, E., Tanca, L.: ADaPT: automatic data personalization based on contextual preferences. In: Proceedings of ICDE 2014, 30th International Conference on Data Engineering, pp. 1234–1237. IEEE Computer Society, USA (2014)
20. Motschnig-Pitrik, R.: A generic framework for the modeling of contexts and its applications. Data Knowl. Eng. **32**(2), 145–180 (2000)
21. Panigati, E.: Personalized management of semantic, dynamic data in pervasive systems: context-addict revisited. In: Proceedings of HPCS 2014, International Conference on High Performance Computing & Simulation, pp. 323–326. IEEE Computer Society, USA (2014)
22. Panigati, E., Rauseo, A., Schreiber, F.A., Tanca, L.: Aspects of pervasive information management: an account of the green move system. In: Proceedings of CSE 2012, 15th International

Conference on Computational Science and Engineering, pp. 648–655. IEEE Computer Society, USA (2012)

23. Quintarelli, E., Rabosio, E., Tanca, L.: A principled approach to context schema evolution in a data management perspective. Inf. Syst. **49**, 65–101 (2015)
24. Raverdy, P.G., Riva, O., de La Chapelle, A., Chibout, R., Issarny, V.: Efficient context-aware service discovery in multi-protocol pervasive environments. In: Proceedings of MDM 2006, 7th International Conference on Mobile Data Management, p. 3. IEEE Computer Society, USA (2006)
25. Roussos, Y., Stavrakas, Y., Pavlaki, V.: Towards a context-aware relational model. In: Proceedings of CRR 2005, International Workshop on Context Representation and Reasoning, pp. 5–8. CEUR-WS.org (2005)
26. Saha, D., Mukherjee, A.: Pervasive computing: a paradigm for the 21st century. IEEE Comput. **36**(3), 25–31 (2003)
27. Sarwat, M., Bao, J., Chow, C.Y., Levandoski, J., Magdy, A., Mokbel, M.F.: Context awareness in mobile systems. In: Francesco, C., et al. (eds.) Data Management in Pervasive Systems. Springer, Heidelberg (2015)
28. Schreiber, F.A., Roveri, M.: Sensors and wireless sensor networks as data sources: models and languages. In: Francesco, C., et al. (eds.) Data Management in Pervasive Systems. Springer, Heidelberg (2015)
29. Schreiber, F.A., Camplani, R., Fortunato, M., Marelli, M., Rota, G.: PerLa: a language and middleware architecture for data management and integration in pervasive information systems. IEEE Trans. Softw. Eng. **38**(2), 478–496 (2012)
30. Schreiber, F.A., Tanca, L., Camplani, R., Vigan`o, D.: Pushing context-awareness down to the core: more flexibility for the PerLa language. In: Proceedings of PersDB 2012, 6th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (2012)
31. Shin, D., won Lee, J., Yeon, J., Lee, S.-g.: Context-aware recommendation by aggregating user context. In: Proceedings of CEC 2009, 11th International Conference on Commerce and Enterprise Computing, pp. 423–430. IEEE Computer Society, USA (2009)
32. Stavrakas, Y., Gergatsoulis, M.: Multidimensional semistructured data: representing context-dependent information on the web. In: Proceedings of CAiSE 2002, 14th International Conference on Advanced Information Systems Engineering, pp. 183–199. Springer, Heidelberg (2002)
33. Stavrakas, Y., Gergatsoulis, M., Rondogiannis, P.: Multidimensional XML. In: Proceedings of DCW 2000, 3rd International Workshop on Distributed Communities on the Web, pp. 100–109. Springer, Heidelberg (2000)
34. Stavrakas, Y., Pristouris, K., Efandis, A., Sellis, T.K.: Implementing a query language for context-dependent semistructured data. In: Proceedings of ADBIS 2004, 8th East European Conference on Advances in Databases and Information Systems, pp. 173–188. Springer (2004)
35. Stefanidis, K., Pitoura, E., Vassiliadis, P.: Managing contextual preferences. Inf. Syst. **36**(8), 1158–1180 (2011)
36. Stoermer, H., Bouquet, P., Palmisano, I., Redavid, D.: A context-based architecture for RDF knowledge bases: approach, implementation and preliminary results. In: Proceedings of RR 2007, 1st International Conference on Web Reasoning and Rule Systems, pp. 209–218. Springer, Heidelberg (2007)
37. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: 1st International Workshop on Advanced Context Modelling, Reasoning and Management (2004)
38. Theodorakis, M., Analyti, A., Constantopoulos, P., Spyratos, N.: A theory of contexts in information bases. Inf. Syst. **27**(3), 151–191 (2002)
39. Yang, S.J.H., Huang, A.F.M., Chen, R.C.S., Tseng, S.S., Shen, Y.S.: Context model and context acquisition for ubiquitous content access in ulearning environments. In: Proceedings of SUTC 2006, International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (vol. 2), pp. 78–83. IEEE Computer Society, USA (2006)
40. Zhang, D., Huang, H., Lai, C.F., Liang, X., Zou, Q., Guo, M.: Survey on context-awareness in ubiquitous media. Multimedia Tools Appl. **67**(1), 179–211 (2013)

Chapter 13
Context Awareness in Mobile Systems

Mohamed Sarwat, Jie Bao, Chi-Yin Chow, Justin Levandoski, Amr Magdy, and Mohamed F. Mokbel

13.1 Overview of Context Awareness in Mobile Systems

Chapter 12 highlights general design methodologies to build context-aware pervasive systems. This chapter, however, presents several techniques that incorporate context awareness in mobile systems. There have been several definitions of context and context awareness (e.g., see [14, 25, 70]). Most of these definitions define the context in terms of examples with special emphasis on the location context. Similarly, there have been several definitions of context-aware applications that include various synonyms, e.g., adaptive applications [13], reactive applications [23], responsive applications [26], situated applications [34], content-sensitive applications [63], and environment-directed applications [29]. This chapter sticks to the most formal definitions given by Dey [24], where context is defined as “any information that can be used to characterize the situation of an entity. An entity

M. Sarwat (✉)
Arizona State University, Tempe, AZ, USA
e-mail: msarwat@asu.edu

J. Bao
Microsoft Research Asia, Beijing, China
e-mail: jiebao@microsoft.com

C.-Y. Chow
City University of Hong Kong, Hong Kong, China
e-mail: chiychow@cityu.edu.hk

J. Levandoski
Microsoft Research, Redmond, WA, USA
e-mail: Justin.Levandoski@microsoft.com

A. Magdy • M.F. Mokbel
University of Minnesota, Minneapolis, MN, USA
e-mail: amr@cs.umn.edu; mokbel@cs.umn.edu

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_13

is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves," while a context-aware system is defined as "A system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

User Context In mobile systems, users may have the ability to specify their personal preferences along with their context. For example, a user may specify that whenever she/is is looking for a restaurant, she/he would like that the query processor takes into account distance, price, rating, and dietary restriction, while when looking for gas stations, the user wants to consider only the distance and the preferred gas company. Using such preferences, the system may consult the appropriate context to provide a context-aware answer that is tailored to both the user preferences and user context. Parts of user context are static, i.e., rarely changing, for example, income, profession, health condition, age, and privacy requirements. Other parts of user context can be highly dynamic, for example, user location which is continuously changing based on user movement.

Point-of-Interest (POI) Context Moreover, points of interest can have their own context as well. Examples of POI data include restaurant databases, hotel databases, and taxi databases. Similar to the concept of user context, POI context may have static and/or dynamic context types. For example, in a restaurant database, a static context may include meal price, dessert price, cuisine, type of people who like this restaurant, and operating hours, while a dynamic context may include the current waiting time, the current number of customers, and today's special.

Environmental Context Context in mobile systems may also come from the environment. Examples of dynamic environmental context include current road traffic (i.e., estimated travel time through road segments), time, weather, and neighboring peers of the query issuer. Examples of static environmental context include third-party ratings of restaurants, gas stations, hotels, or any other third-party statistics about data sources. Examples of queries that get help from the environmental context include a restaurant finder query where the application consults the environmental context to get the current traffic and estimated travel time to each restaurant and, then, the computed distance is fed to the query to get an accurate answer. Another example that makes use of the data collected by third parties is also a restaurant finder query in which the user wants to choose the best restaurant in terms of other user reviews and scores for each restaurant.

Incorporating such contextual data in services provided to mobile users may significantly enhance the quality of service in terms of finding more relevant answers. This chapter presents different ways of expressing the spatial location context within the service. Then, the chapter discusses three main application examples that can take advantage of knowing various contexts, namely, social networking (Sect. 13.2.1), microblogging platforms (Sect. 13.2.2), and recommendation services (Sect. 13.2.3). The chapter finally presents a generic method that incorporates context and user preference awareness in database systems (Sect. 13.3)—which may serve as a back end for context-aware mobile systems.

13.2 Spatial Location as a Context

One prevalent example of context is a user's location. In this vein, location-based systems take into account the user's location context when providing query answers. Location-based services are viewed as the convergence of technologies of mobile devices, GIS/spatial databases, and the Internet. Location-based services aim to provide new services to their users based on the awareness of their locations. Examples of these services include continuous live traffic reports (*Continuously, let me know if there is congestion within five minutes of my route*), emergency response (*Dispatch the nearest five police cars to the crime scene*), safety assurance (*Always ensure that there are at least k ambulances within certain areas of interest*), and store finder (*Where is my nearest restaurant*?). The flood of information that comes out from location-detection devices and the large number of mobile users who utilize location-based services call for migrating the functionality of location-based services with recent technologies into database management systems (DBMSs). The main idea is to model user requests for location-based services as location-based queries (i.e., spatiotemporal queries) that can be efficiently executed over large numbers of mobile users through database management modules, e.g., data indexing, query processing, and query optimization.

Several systems have been developed over the years to provide database support for location-based queries. These systems include DOMINO [84], SECONDO [30], and PLACE [55, 57]. The DOMINO system [84] provides several location-based features on top of existing DBMSs including dynamic attributes, a spatial and temporal query language, indexing, and uncertainty management. SECONDO [30] is an extensible database system built to support a plethora of nonstandard applications, e.g., location-based services, through algebra modules. The PLACE server [55, 57] provides the first approach that embedded location-based support inside a DBMS through specialized location-based query operators. The rest of this section presents techniques that incorporate location awareness into modern social networks, microblogging platforms, and recommendation services.

13.2.1 Location-Aware Social Networking

Social networking systems, e.g., Facebook and Twitter, and news aggregators, e.g., iGoogle and My Yahoo!, are among the most popular Web services nowadays [22]. The most important functionality shared by such Web services is the *news feed*, where the users can post personal messages or news to be received by friends or subscribers. Moreover, nowadays, many social networking systems allow the user to attach locations to their messages, e.g., Facebook Places and Twitter Nearby. These existing systems either treat the message location as an additional tag, e.g., Facebook Place, or a point indicating the message issuing location, e.g., Twitter Nearby [67]. However, the spatial information of many messages posted in the social networking systems or news aggregators should be a spatial extent rather than a point location. For example, in the social networking context, a user may announce a party using her/him neighborhood as the spatial context. Similarly, in the news aggregator context, the weather news agent may post local weather updates for an area, or a traffic news agent posts local traffic conditions for a set of highways. The users of such systems find these messages relevant only if their locations are within the message spatial contexts. Considering the DATABENC project case study, a museum in Naples may announce a special historical exhibition (event) on its Facebook page (through a Facebook message). Only users (i.e., Naples tourists) who are within the range of such a message will benefit from this announcement.

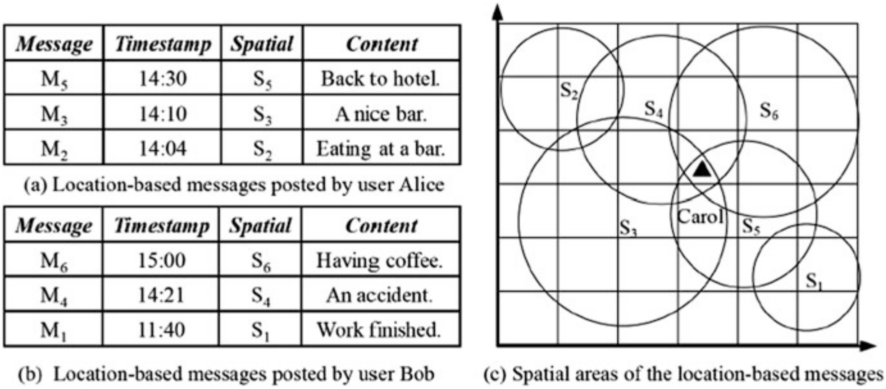
13.2.1.1 Location-Aware News Feeds

Directly applying a spatial filter over the existing news feed system, e.g., [71], is extremely inefficient. To this end, novel location-based news feed systems are proposed, embedding the spatial pruning techniques deep inside the location-aware news feed functionality and improving the system efficiency significantly. This section presents two main methods to process location-aware news feed: (1) GeoFeed [11], which efficiently support users' location-aware news feed requests by considering their activity patterns, and (2) MobiFeed [86], which considers the user's traveling patterns in delivering the news feeds.

GeoFeed is the first attempt of building a location-aware news feed system. GeoFeed enables the user to post messages with spatial extents indicating the spatial relevance and also to get the news feed that is recent and spatially relevant to their current locations from their friends or favorite news agents. GeoFeed abstracts the location-aware news feed functionality for a user u to a set $\mathcal{Q}_u$ of location-based point queries posed to the user's friends $\mathcal{F}_u$. Each query $q_i \in \mathcal{Q}_u$ is posed to a friend $f_i \in \mathcal{F}_u$ to retrieve the k most recent spatially relevant messages to u's location. Then, u may opt to get all the received messages or add a filter to select only the k most recent and spatially relevant messages from *all* users. As an example, consider user Carol, depicted by a triangle in Fig. 13.1c, which is a friend of both users Alice and Bob. Carol issues two location-based point queries, with $k = 2$, one for Alice and one for Bob. The query to Alice returns M_3 and M_5, while the query to Bob returns M_4 and M_6, as Carol's location lies inside the spatial extents of these four messages. Carol can add an additional filter to get the two most recent messages, i.e., M_5 and M_6.

As a result, for each location-aware news feed request, $|u.\mathcal{F}|$ location-based point queries will be executed to user's friends. GeoFeed extends the traditional news feed query methods, *pull and push*, spatial aware, to address these point queries:

Spatial Pull Approach Spatial pull approach answers location-based point query q_i by exploiting a spatial index.



Message	*Timestamp*	*Spatial*	*Content*
M_5	14:30	S_5	Back to hotel.
M_3	14:10	S_3	A nice bar.
M_2	14:04	S_2	Eating at a bar.

(a) Location-based messages posted by user Alice

Message	*Timestamp*	*Spatial*	*Content*
M_6	15:00	S_6	Having coffee.
M_4	14:21	S_4	An accident.
M_1	11:40	S_1	Work finished.

(b) Location-based messages posted by user Bob

(c) Spatial areas of the location-based messages

Fig. 13.1 Location-based messages and news feed

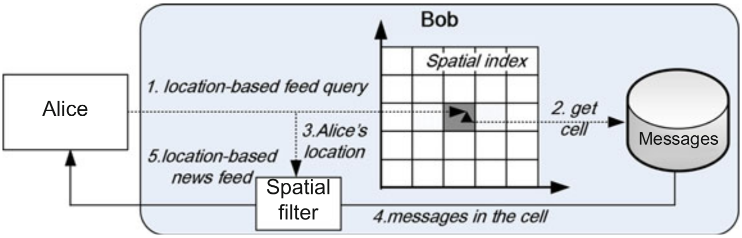


Fig. 13.2 Example of spatial pull approach

Figure 13.2 gives an example of the main idea of the *spatial pull* approach where user *Alice* needs to get her k spatially relevant messages from her friend *Bob*. The execution flow goes as follows: (1) *Alice* submits her location and the news feed query to *Bob*. (2) *Bob* exploits his grid index structure $\mathcal{G}_{Bob}$ to find out cell C that includes *Alice*'s location and retrieves all the messages stored in C. (3) *Bob* applies a spatial filter over all the messages returned from $\mathcal{G}_{Bob}$ to only report those messages that include *Alice*'s location (depicted by a black triangle), as there could be messages in C that do not overlap with *Alice*'s location. (4) If more than k messages are returned from the spatial filter, *Bob* forwards the k most recent ones to *Alice* as part of her requested location-aware news feed. *Alice* will get the rest of her news feed from her other friends.

The advantage of this approach is that the system does not have any overhead, when the user is offline. However, considering the number of location-based queries in a news feed request, the response time is significant, when the user has a large number of friends.

Spatial Push Approach The *spatial push* approach in GeoFeed precomputes and stores the answer of the location-based news feed query in a materialized view maintained by the friend f_i. Then, once the user u logs on, u only retrieves the news feed from the materialized view.

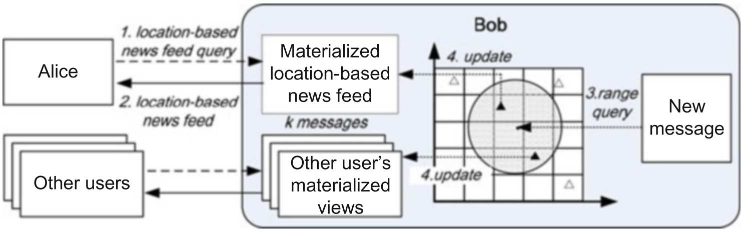

Alice
1. location-based news feed query
2. location-based news feed
Bob
Materialized location-based news feed
k messages
4. update
3.range query
New message
Other users
Other user's materialized views
4.update

Fig. 13.3 Example of spatial push approach

Figure 13.4 gives an example of the main idea of the *spatial push* approach in GeoFeed where user *Alice* needs to get her *k* spatially relevant messages from *Bob*. The *spatial push* approach consists of two orthogonal parts, namely, *query processing* and *view maintenance*, detailed as follows:

1. *Query processing.* As *Alice*'s news feed query posed to *Bob* uses the *spatial push* approach, *Alice* registers her location with the grid index structure maintained by *Bob*, $\mathscr{G}_{Bob}$. In turn, *Bob* deals with *Alice*'s location as a continuous location-based point query [19, 56, 60] that needs to be maintained, in a materialized view, even if *Alice* goes offline. Once *Alice* logs on to GeoFeed, she just probes her materialized view, maintained by *Bob*, to retrieve her news feed.
2. *View maintenance.* The friend *Bob* uses his grid index structure, $\mathscr{G}_{Bob}$, to maintain a materialized view for each user employing the *spatial push* approach to retrieve his/her messages from him. Any new message *M* produced from *Bob* generates a range query over the grid index, $\mathscr{G}_{Bob}$, to retrieve the locations of all friends of *Bob* who are located within the spatial range of *M* and use the *spatial push* approach to retrieve their news feed from *Bob*. For each of these friends, *M* is forwarded to the designated materialized view. Figure 13.3 shows *Alice*'s location in $\mathscr{G}_{Bob}$ as a black triangle. The message *M* produced from *Bob* is depicted as a shaded circle over $\mathscr{G}_{Bob}$, which updates the materialized view of *Alice* among other views of those users who retrieve their news feed from *Bob* with the *spatial push* approach. Although the *spatial push* approach is very appealing to the user, it poses a large overhead over the system resources to continuously maintain the materialized view while the user is offline.

Shared Push Approach The *shared push* approach in GeoFeed is designed to take advantage of the users' locality to reduce the system overhead of the *spatial push* approach while only slightly increasing the response time of location-based news feed queries. The main idea of the *shared push* approach is to maintain one materialized view shared among the different users or different log-in locations from the same user within one cell.

Figure 13.4 gives an example of how *Alice* gets her *k* relevant messages from her friend *Bob*, with the *shared push* approach. The scenario is very similar to the

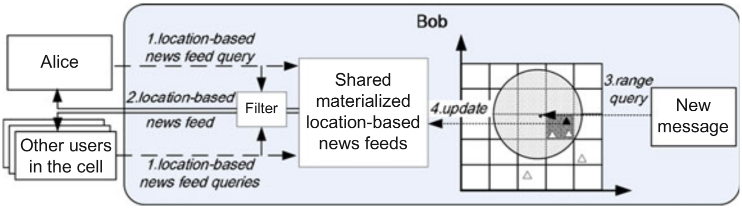

Bob
Alice
1.location-based news feed query
2.location-based news feed
Filter
Shared materialized location-based news feeds
Other users in the cell
1.location-based news feed queries
4.update
3.range query
New message

Fig. 13.4 Example of shared push approach

spatial push approach with the following two differences in query processing and view maintenance:

1. *Query processing.* As several users share the same materialized view with *Alice*, *Bob* needs to add an additional filter after the returned answer from the shared materialized view to filter out those news items that are not relevant to *Alice*.
2. *View maintenance.* Instead of maintaining one materialized view for each friend of *Bob* as in the *spatial push* approach, the *shared push* approach maintains one shared materialized view across all the users with the *spatial push* approach in the cell. Thus, a new message coming out from *Bob* will be inserted in much less materialized views.

GeoFeed builds a cost model to quantify the cost of each approach based on different user activity patterns, i.e., offline period and message update frequency. Finally, GeoFeed proposes a decision algorithm to choose the best approach to evaluate the location-based point queries for each user-friend connection. The main objectives of the decision algorithm are the following: (1) guarantee a fair user response time for the users to get their news feed and (2) reduce the overall system overhead.

Therefore, the GeoFeed decision model consists of three main steps: (1) Step 1, *response time guarantee*, finds out the maximum number of *spatial pull* queries, $NQPull_u$, that user u can afford while having the news feed within $\mathscr{T}_u$; (2) Step 2, *pull vs. push selection*, decides on what are these $NQPull_u$; out of all the queries posed by u that will be assigned to the *spatial pull* approach, other queries will be assigned to the *spatial push* approach; and (3) Step 3, *refinement with shared push*, finds out if using the *shared push* approach for any grid cell C maintained at friend f_i can reduce GeoFeed system overhead.

13.2.1.2 Mobility-Aware News Feed

MobiFeed is a location-aware news feed framework specially designed for *moving* users. As shown in the application scenario in Fig. 13.5, MobiFeed enables a mobile user, say Alice, to post a message and tag a point (e.g., M_1), a spatial extent (e.g., M_{14} is associated with a circular spatial area), or a venue (e.g., M_6 and M_7 are spatially associated with restaurant R_1) as its geo-location. Alice can also issue a location-

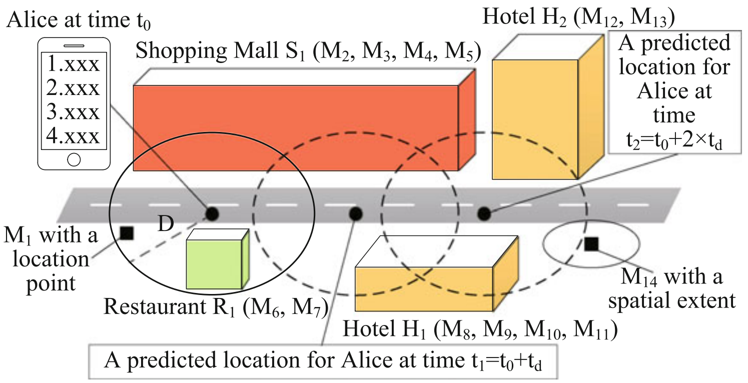

Alice at time t0
1.xxx
2.xxx
3.xxx
4.xxx
Shopping Mall S1 (M2, M3, M4, M5)
Hotel H2 (M12, M13)
A predicted location for Alice at time t2=t0+2×td
D
M1 with a location point
Restaurant R1 (M6, M7)
Hotel H1 (M8, M9, M10, M11)
M14 with a spatial extent
A predicted location for Alice at time t1=t0+td

Fig. 13.5 An application scenario of MobiFeed

aware news feed query to receive nearby user-generated messages, e.g., “Alice can receive 4 messages that are the most relevant to her among the messages within 1 km from her location every 10 seconds” (Fig. 13.5).

To deal with such a moving scenario, one possible solution is to adjust GeoFeed by selecting k most relevant messages as a news feed for each query region. However, this solution may not optimize the *overall* quality of news feeds. This is because the news feeds are only computed based on a user’s location at the query time (i.e., it does not consider the user’s future locations). For example, in Fig. 13.5, there are 14 messages (i.e., M_1 to M_{14}) with their geo-location intersecting Alice’s query regions at time t_0, t_1, and/or t_2. Assume M_i is more relevant to Alice than M_j if $i < j$. GeoFeed returns (M_1, M_2, M_3, M_4) at t_0, (M_5, M_8, M_9, M_{10}) at t_1, and $(M_{11}, M_{12}, M_{13}, M_{14})$ at t_2. However, given Alice’s location at t_0, if we predict two future locations at t_1 and t_2 and consider all three query regions at the same time, a better solution returns (M_1, M_2, M_6, M_7) at t_0, (M_3, M_4, M_5, M_8) at t_1, and $(M_9, M_{10}, M_{11}, M_{12})$ at t_2 because M_6 and M_7 are more relevant to Alice than M_{13} and M_{14}.

To this end, MobiFeed incorporates location prediction to the process of location-aware news feed generation, thus formulating a novel n-look-ahead news feed scheduling framework to improve the overall quality of *multiple* news feeds for moving users. Figure 13.6 depicts an overview of the MobiFeed framework. MobiFeed stores geo-tagged user-generated messages in a database, and it consists of three key functions: *location prediction*, *relevance measure*, and *news feed scheduler*. As shown in Fig. 13.5, given a user u’s location $u.location$ at current time t_0, u’s required minimum message display time t_d, u’s specified range distance D, u’s requested number of messages per news feed k, and a look-ahead step n, the *location prediction* function estimates n future locations for u at times $t_1 = t_0 + t_d$, $t_2 = t_0 + 2 \times t_d, \ldots,$ and $t_n = t_0 + n \times t_d$; the *relevance measure* function calculates the relevance score of each candidate message whose geo-location intersects any u’s query region (i.e., a circular area centered at $u.location$ or a predicted location with

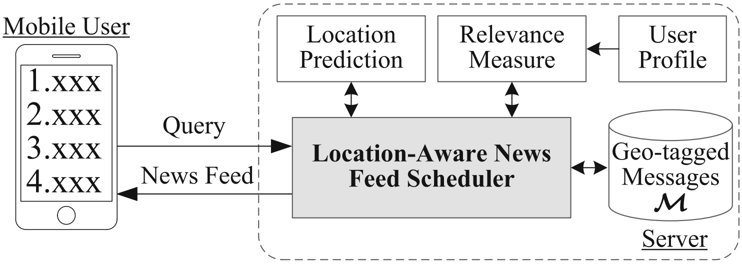


Fig. 13.6 MobiFeed framework

a radius D); and the *news feed scheduler* generates news feeds from the candidate messages for u's query regions at t_0, t_1, . . ., and t_n with the highest overall relevance score. The computed $n + 1$ news feeds are sent to u. u's mobile device immediately displays the first news feed, i.e., the one with respect to the query region at t_0, and then displays each of the remaining news feeds one by one for every t_d. Generally, MobiFeed is equipped with a location-aware *news feed scheduler*, which works with the *location prediction* and *relevance measure* functions to provide high-quality news feeds for moving users. These three functions are illustrated below:

Location Prediction The *location prediction* function can employ any existing location prediction algorithm if it can predict a user's location at a specified future time in a road network. As an implementation, MobiFeed incorporates a path prediction algorithm [35] into MobiFeed. This algorithm represents the mobility statistics (e.g., direction, speed) extracted from historical user trajectories and predicts a user's future location using a maximum likelihood method.

Relevance Measure To indicate the relevance of a message M_j to a user u_i, GeoRank considers both the temporal factor and the spatial factor. Users have preferences over messages in terms of their categories and contents. To this end, in order to return a score to indicate the relevance of a message M_j to a user u_i, i.e., relevanceScore(u_i, M_j), the *relevance measure* function considers three factors: (a) Message categories. For each user u_i in the *user profile* (see Fig. 13.6), the system maintains a list of categories $CateList_i$ sorted by the number of u_i's submitted messages belonging to each category. (b) Message contents use the vector space model to calculate the similarity between M_j and u_i's submitted messages in terms of their contents. (c) Spatial proximity measures the Euclidean distance between M_j and u_i, where closer messages are favored. The system employs two-level and linear combinations to integrate the aforementioned three factors into the *relevance measure* function. In the first level, the system selects top-σ categories $u_i.C$ in $CateList_i$ for a querying user u_i, where σ is a system parameter to specify the number of top categories. All messages within u_i's range distance $u_i.D$ and belonging to categories in $u_i.C$ are considered as the set of candidate messages. At the second level, for each candidate message, the system measures its relevance to u_i using a linear combination of (b) and (c); a user-defined preference parameter $0 \leq \beta \leq 1$

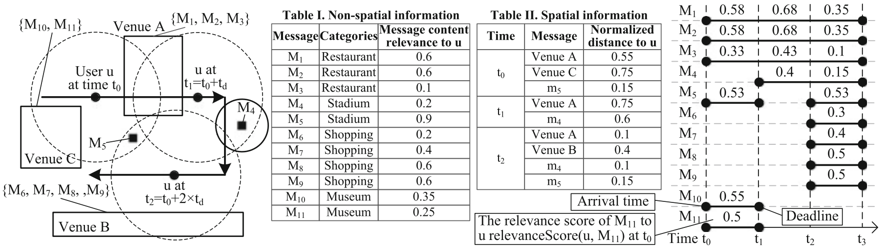


Table I. Non-spatial information

Message	Categories	Message content relevance to u
M_1	Restaurant	0.6
M_2	Restaurant	0.6
M_3	Restaurant	0.1
M_4	Stadium	0.2
M_5	Stadium	0.9
M_6	Shopping	0.2
M_7	Shopping	0.4
M_8	Shopping	0.6
M_9	Shopping	0.6
M_{10}	Museum	0.35
M_{11}	Museum	0.25

Table II. Spatial information

Time	Message	Normalized distance to u
t_0	Venue A	0.55
	Venue C	0.75
	m_5	0.15
t_1	Venue A	0.75
	m_4	0.6
t_2	Venue A	0.1
	Venue B	0.4
	m_4	0.1
	m_5	0.15

Fig. 13.7 Location-aware news feed scheduling

indicates the importance of the spatial proximity factor with respect to the message content factor. In our running example (Fig. 13.7) where $\delta = 4$, $n = 2$, and $\beta = 0.5$, there are 11 candidate messages (i.e., M_1 to M_{11}) intersecting three query regions, and these messages all belong to the top 4 categories (i.e., restaurant, stadium, shopping, and museum). For user u and each candidate message M_j, the message content score and spatial proximity score are listed in Tables I and II, respectively. At the second level, the system calculates the relevance score of candidate messages for the query regions at times t_0, t_1, and t_2, and the result is shown in the timeline chart in Fig. 13.7.

News Feed Scheduler With $n + 1$ sets of candidate messages along with their relevance scores, the *news feed scheduler* aims at scheduling at most $k \times (n + 1)$ candidate messages to the $n + 1$ news feed, such that the overall relevance score of these news feeds is maximized. First, the system illustrates how to qualify the news feeds. In information retrieval, query-relevance ranking algorithms usually display a document that is more relevant to a user's query at a higher position in a result list [10]. To this end, MobiFeed supports different weights for different positions in a news feed result list, i.e., a higher weight is given to a message displayed at a higher position because it would be easier to draw a user's attention. MobiFeed uses a simple weighting scheme. Given a result list with k positions, the weight of the first position is k, the weight of the second position is $k - 1$, and so on. In general, the weight of a message M_j at the jth position ($1 \leq j \leq k$) is $displayWeight(j, k) = k - (j - 1)$. Thus, the relevance score of a news feed f_i with k messages $M_1, M_2, \ldots, M_k$ displayed at the jth position in a result list for a user u_i is calculated as follows: $relevanceScore(f_i) = \sum_{j=1}^{k} relevanceScore(u_i, M_j) \times displayWeight(j, k)$, where $relevanceScore(u_i, M_j)$ is the score returned by the *relevance measure* function. Finally, the total relevance score of $n + 1$ news feeds is the sum of their relevance scores.

As depicted in the running example (Fig. 13.7), a geo-tagged message may be included in multiple sets of candidate messages. For example, M_1 is included in all three sets of candidate messages, so M_1 can be scheduled to one of the news feeds at t_0, t_1, and t_2, or none of them. This step aims at generating $n + 1$ news feeds such that their overall relevance score is maximized. For each query region q_i, the system calculates a score for its first candidate message (i.e., the message with the highest relevance score) M_j by $relevanceScore(u, M_j) \times displayWeight(j, k)$, where u is the querying user and j is the highest available position in q_i's news feed result list. The message with the largest such score, denoted as *BestMsg*, is assigned to the result list of the corresponding query region. Then, the scheduler removes *BestMsg* from all the candidate message sets. Candidate messages are iteratively selected for appropriate query regions until the news feed in each query region has k messages or all candidate message sets become empty. Whenever k messages have been assigned to the result list of a query region, its corresponding candidate message set is discarded. The computed $n + 1$ news feeds are sent to u.

13.2.2 Location-Aware Microblogging

Microblogging platforms, e.g., Twitter, Facebook timelines, Foursquare, and Sina Weibo, have become incredibly popular among Web users in the last few years. Every day, 271+ million active Twitter users generate 500+ million tweets [79, 80], while 1.23+ billion Facebook users post 3.2+ billion comments [27]. Microblogs data is rich and carries different types of information including text, location information, and users' information. Moreover, microblogs textual content is rich with user updates on real-time events, interesting keywords/hashtags, news items, opinions/reviews, hyperlinks, images, and videos. This richness of microblogs data and the continuous arrival as a data stream have triggered the need for the research community to tackle a new set of emerging queries and applications that were not applicable before on traditional data streams. For example, keyword search on streaming data [16, 18, 85, 87] is firstly introduced on microblogs. Other newly emerging applications on microblogs include event detection [1, 48, 54, 65, 82], news extraction [3, 59, 66], location-aware search [53], and analysis [31, 77, 78]. Such kinds of applications are so important that major IT companies spend millions of dollars to enable them to their customers [7, 58].

As user-generated data, microblogs data contains inherent user context information including location, social bonds and interactions, interest group affiliations, time zone, and language. Particularly, location context information is the most important and usable in the microblogs research literature [1, 15, 32, 48, 50–53, 66, 73, 74, 82]. This comes from the plethora of location information where 79 % of Facebook active users and 76 % of Twitter active users are mobile users, while Foursquare services are all location-aware. With the advances of GPS-equipped mobile devices, this leads to an explosive richness in the availability of location information that comes with microblogs. This availability of microblogs locations enables very important location-aware services and applications like real-time news delivery [3], rescue services during natural disasters [72], and geo-targeted advertisements[1]. In the context of the DATABENC case study, a tourist may want to browse geo-tagged tweets that are located in Naples.

[1] New Enhanced Geo-targeting for Marketers. https://blog.twitter.com/2012/new-enhanced-geo-targeting-for-marketers

Location awareness in microblogs applications mainly lies in two broad categories: *offline* and *real-time* (or *online*) location-aware applications. Offline applications exploit microblogs locations in a non-streaming fashion to provide, for example, effective visualization [54], news extraction [66], and geo-correlation modeling [32]. These applications mostly depend on exploiting the new kind of data with existing spatial technologies to support new kinds of services and analyses. Due to the real-time sensitive contents of microblogs [3, 72], the newly active and emerging area on microblogs is the real-time location awareness. Among those real-time applications are local event detection [1, 65, 82], geo-correlated trend discovery [15], local frequent keyword search [74], location-aware search [53], and arbitrary analysis queries [51, 77, 78]. Each of these applications works with either a *continuous query* paradigm or an *indexed stream* paradigm. In a continuous query paradigm, microblogs are processed once at their arrival and then discarded from the system. The users are continuously provided with incremental output through the application interface, e.g., local events on a geographical map. This paradigm is similar to processing continuous queries on traditional data streams. This usually provides very low memory footprint along with high digestion rates for incoming microblogs. On the contrary, the indexed stream paradigm digests incoming microblogs in index data structures that are suitable for the supported queries. When a query comes at user will, the indexed data are processed to get the query answer. This paradigm mainly serves snapshot queries which are very popular on microblogs. However, indexing fast-streaming data introduces new challenges in terms of digestion rates and memory footprint. Thus, the applications that use the indexed stream paradigm have to address three main challenges: scalable data digestion, efficient memory utilization, and fast query processing.

To exploit microblogs locations in an indexed stream paradigm, various spatial real-time indexing techniques are explored in the literature [15, 53, 74] to support different queries. The common theme among all these techniques is to reduce the indexing latency and increase the digestion rate. Consequently, all of them use main-memory index structures for fast data digestion and retrieval. In addition, the in-memory indexes are carefully tuned to provide the minimal indexing overhead. To this end, space-partitioning spatial indexes are used to ingest real-time microblogs, e.g., spatial grid index or spatial quad-tree. Contrary to data-partitioning spatial indexes, e.g., R-tree, cell boundaries in space-partitioning indexes do not change with the incoming data. Instead, each cell covers a specific area of the space and holds whatever data it contains. This significantly reduces the overhead of restructuring index cells in real time. In addition, microblogs indexes are equipped with efficient update and restructuring techniques. Real-time insertion in the index comes in batches so that the amortized insertion cost per microblog is minimized. Also, deletions are performed on a lazy basis, when cells are accessed for insertion, so that the cost of traversing index cells is piggybacked on the insertion cost. When index cells are over- or underutilized, they are not split or merged eagerly. Instead, lazy split and merge operations have shown to significantly reduce the restructuring overhead. In brief, spatial indexing of the microblogs stream in real time makes use of space-partitioning indexes, batch insertions, lazy deletions, and lazy index restructuring to reduce the indexing overhead and increase the digestion rates.

A spatial pyramid index is a commonly used space-partitioning index in handling microblogs data in real time. This index maintains multiple levels, each of them is a spatial grid index that consists of fixed-sized spatial cells. The cells of a level are spatially disjoint from those of the other levels. The deeper the level, the finer the granularity and so the smaller the cell size. The pyramid index is used in two forms: either in a complete pyramid form, e.g., in [74], or in a partial pyramid form, e.g., in [53]. In a complete pyramid, each level is a complete grid with all cells maintained. On the contrary, each level of the partial pyramid is an incomplete grid that maintains only a subset of its cells. Each form is used to handle certain application requirements. For example, to efficiently handle data spatial skewness, the partial pyramid index is used where the regions with more dense data dynamically span deep levels of the index, while sparse regions are represented only in high levels. Another example is processing queries on arbitrary spatial regions with only a few number of cells. In this case, a complete pyramid is used and the data is replicated across all levels so that any small or large spatial region could be represented with the minimum number of cells. In a nutshell, the core idea is to use fixed-sized spatial cells that are efficient to maintain in real time and organize them in different ways to handle different requirements.

As main-memory indexes, microblogs indexing techniques are required to efficiently utilize main-memory resources. To optimize for memory consumption, naturally, the different techniques behave differently based on the supported queries. The main optimization line is to store only necessary information in main memory, e.g., recent microblogs or local top frequent keywords, and get rid of any other information so that incoming queries can barely find its answers in the main memory. This allows to serve queries with larger time horizons and low response times.

The aforementioned techniques tackled location awareness in microblogs in terms of individual applications that could exploit the location information in offline or online fashion. However, with the increasing demand on microblogs data and applications, the upcoming microblogs data research would go toward end-to-end management systems that would involve a long-history keeping to support different types of applications as well as complex analysis queries over long periods of time [51, 77, 78]. For example, political events like elections or protests are kept active for several months on social media which requires management of long-history data as well as recent data. For such kind of systems and applications, microblogs would not be managed only under the *indexed stream* paradigm, which is mainly used to manage recent streaming data in main memory, but also would need a form of disk-based offline indexing and a flushing management from main memory to disk. For the offline microblogs management, existing big data platforms [6] and technologies are expected to scale for the large volume of microblogs data. Yet, the new systems would be required to research the suitable flushing policies that would work as the glue between memory-based and disk-based data management components.

13.2.3 Location-Aware Recommender Systems

Recommender systems [2, 47, 68] (introduced in Chap. 11) make use of community opinions to help users identify useful items from a considerably large search space (e.g., Amazon inventory [49], Netflix movies[2]). The technique used by many of these systems is collaborative filtering (CF) [64], which analyzes past community opinions to find correlations of similar users and items to suggest k personalized items (e.g., movies) to a querying user u. Community opinions are expressed through explicit ratings represented by the triple (*user*, *rating*, *item*) that represents a *user* providing a numeric *rating* for an *item*.

Currently, myriad applications can produce *location-based ratings* that embed user and/or item locations. For example, location-based social networks (e.g., Foursquare[3] and Facebook Places[4]) allow users to "check in" at spatial destinations (e.g., restaurants) and rate their visit; thus, they are capable of associating both user and item locations with ratings. Such ratings motivate an interesting new paradigm of *location-aware recommendations*, whereby the recommender system exploits the spatial aspect of ratings when producing recommendations. For instance, in light of the DATABENC project case study, location-aware recommender systems (LARS) may suggest museums in Naples to tourists based on their spatial locations. Existing recommendation techniques [2] assume ratings are represented by the (*user*, *rating*, *item*) triple; thus, they are ill-equipped to produce location-aware recommendations.

13.2.3.1 A Study of Location-Based Ratings

To motivate location-aware recommendation services, this section presents an analysis of two real-world location-based rating datasets [46]: (1) a subset of the well-known MovieLens dataset[5] containing approximately 87 K movie ratings associated with user zip codes (i.e., spatial ratings for nonspatial items) and (2) data from the Foursquare[6] location-based social network containing user visit data for 1M users to 643 K venues across the United States (i.e., spatial ratings for spatial items). In our analysis one can consistently observe two interesting properties that motivate the need for location-aware recommendation techniques.

Preference Locality Preference locality suggests users from a spatial region (e.g., neighborhood) prefer items (e.g., movies, destinations) that are manifestly different than items preferred by users from other, even adjacent, regions. Figure 13.8a shows lists of the top 4 movie genres using average MovieLens ratings of users from

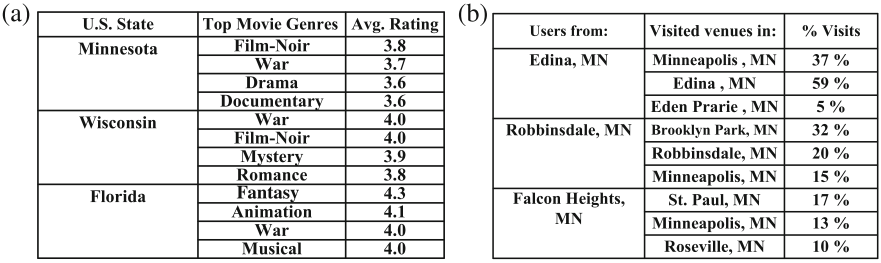
[2] Netflix: http://www.netflix.com

[3] Foursquare: http://foursquare.com

[4] The Facebook Blog, "Facebook Places". http://tinyurl.com/3aetfs3

[5] MovieLens. http://www.movielens.org/

[6] http://foursquare.com



(a)

U.S. State	Top Movie Genres	Avg. Rating
Minnesota	Film-Noir	3.8
	War	3.7
	Drama	3.6
	Documentary	3.6
Wisconsin	War	4.0
	Film-Noir	4.0
	Mystery	3.9
	Romance	3.8
Florida	Fantasy	4.3
	Animation	4.1
	War	4.0
	Musical	4.0

(b)

Users from:	Visited venues in:	% Visits
Edina, MN	Minneapolis , MN	37 %
	Edina , MN	59 %
	Eden Prarie , MN	5 %
Robbinsdale, MN	Brooklyn Park, MN	32 %
	Robbinsdale, MN	20 %
	Minneapolis, MN	15 %
Falcon Heights, MN	St. Paul, MN	17 %
	Minneapolis, MN	13 %
	Roseville, MN	10 %

Fig. 13.8 Preference locality in location-based ratings. (**a**) MovieLens preference locality. (**b**) Foursquare preference locality

different US states. While each list is different, the top genres from Florida differ vastly from the others. Florida's list contains three genres ("fantasy," "animation," "musical") not in the other lists. This difference implies movie preferences are unique to specific spatial regions and confirms previous work from the New York Times[7] that analyzed Netflix user queues across US zip codes and found similar differences. Meanwhile, Fig. 13.8b summarizes our observation of preference locality in Foursquare by depicting the visit destinations for users from three *adjacent* Minnesota cities. Each sample exhibits diverse behavior: users from Falcon Heights, MN, favor venues in St. Paul, MN (17 % of visits), Minneapolis, MN (13 %), and Roseville, MN (10 %), while users from Robbinsdale, MN, prefer venues in Brooklyn Park, MN (32 %), and Robbinsdale, MN (20 %). Preference locality suggests that recommendations should be influenced by location-based ratings *spatially close* to the user. The intuition is that localization influences recommendation using the unique preferences found within the spatial region containing the user.

Travel Locality Our second observation is that when recommended items are spatial, users tend to travel a limited distance when visiting these venues. This chapter refers to this property as "travel locality." In Foursquare data analysis, 45 % of users travel 10 miles or less, while 75 % travel 50 miles or less. This observation suggests that spatial items closer in travel distance to a user should be given precedence as recommendation candidates. In other words, a recommendation loses efficacy the further a querying user travels to visit the destination. Existing recommendation techniques do not consider travel locality and thus may recommend user's destinations with burdensome travel distances (e.g., a user in Chicago receiving restaurant recommendations in Seattle).

[7] New York Times - A Peek Into Netflix Queues. http://www.nytimes.com/interactive/2010/01/10/nyregion/20100110-netflix-map.html

13.2.3.2 Location-Aware Collaborative Filtering Recommender Systems

This section describes a system that incorporates location awareness in collaborative filtering, namely, LARS (location-aware recommender system) [69]. Like traditional recommender systems, LARS suggests k items personalized for a querying user u. However, LARS is distinct in its ability to produce location-aware recommendations using *each* of the following taxonomy of *three* types of location-based ratings:

1. Spatial ratings for nonspatial items, represented as a four-tuple (*user*, *ulocation*, *rating*, *item*), where *ulocation* represents a user location. Items are nonspatial in nature (e.g., movies) and thus do not have an associated location. As an example, traditional e-commerce applications (e.g., Netflix) may use a user's home address as ulocation, while mobile applications may associate the location where the user rated the item as the ulocation.
2. Nonspatial ratings for spatial items, represented as a four-tuple (*user*, *rating*, *item*, *ilocation*), where *ilocation* represents an item location. Examples of applications that produce such ratings are e-commerce applications that gather user opinions on venues/destinations (e.g., restaurant review websites), but do not collect user locations.
3. Spatial ratings for spatial items, represented as a five-tuple (*user*, *ulocation*, *rating*, *item*, *ilocation*), for example, a user at his/her office rating a restaurant that he/she visited for lunch.

LARS produces recommendations using *spatial ratings for nonspatial items*, i.e., the tuple (*user*, *ulocation*, *rating*, *item*), by employing a *user partitioning* technique that exploits preference locality. This technique uses an adaptive pyramid structure (see Fig. 13.9) to partition ratings by their *user location* attribute into spatial regions of varying sizes at different hierarchies. For a querying user located in a region R, the system applies an existing collaborative filtering technique that utilizes

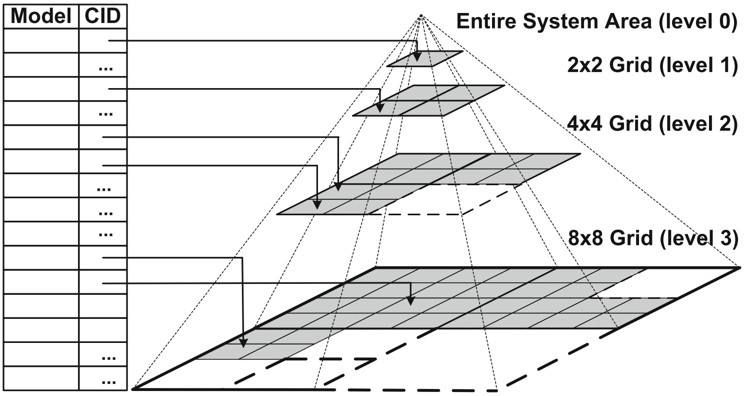


Fig. 13.9 Partial pyramid structure

only the ratings located in *R*. The challenge, however, is to determine whether all regions in the pyramid must be maintained in order to balance two contradicting factors: *scalability* and *locality*. Maintaining a large number of regions increases *locality* (i.e., recommendations unique to smaller spatial regions) and yet adversely affects system *scalability* because each region requires storage and maintenance of a collaborative filtering data structure necessary to produce recommendations (i.e., the recommender model). The LARS pyramid dynamically adapts to find the right pyramid shape that balances scalability and recommendation locality.

LARS produces recommendations using *nonspatial ratings for spatial items*, i.e., the tuple (*user*, *rating*, *item*, *ilocation*), by using *travel penalty*, a technique that exploits travel locality. This technique penalizes recommendation candidates the further they are in travel distance to a querying user. The challenge here is to avoid computing the travel distance for all spatial items to produce the list of *k* recommendations, as this will greatly consume system resources. LARS addresses this challenge by employing an efficient query processing framework capable of terminating early once it discovers that the list of *k* answers cannot be altered by processing more recommendation candidates. To produce recommendations using *spatial ratings for spatial items*, i.e., the tuple (*user*, *ulocation*, *rating*, *item*, *ilocation*), LARS employs both the *user partitioning* and *travel penalty* techniques to address the user and item locations associated with the ratings. This is a salient feature of LARS, as the two techniques can be used separately, or in concert, depending on the location-based rating type available in the system.

13.2.3.3 Non-collaborative Filtering Location-Aware Recommenders

Inferring how a certain user would like (rating) a certain location is very challenging using a user's location history. First, a user can only visit a limited number of physical locations. This results in a sparse user location matrix for most existing location recommendation systems which directly play a collaborative filtering-based model over physical locations. Second, the task becomes even more difficult when an individual travels to a new place (see Figure 13.10: User Location History Distributions), where she/he has visited few locations. For example, Fig. 13.10a, b plots the locations (according to the tips in Foursquare) visited by people from New York City, in Los Angeles (LA) and New York City (NYC), respectively. Clearly, the tip records generated by NYC people are very few in LA, which are only 0.47 % of the records they left in NYC and 0.75 % of the records generated by local users in LA. This phenomenon is quite common in the real world, aggravating the data sparsity problem to location rating inference (if we want to provide people from NYC with location recommendations in LA). In this case, solely using a CF model is not feasible any more. First, one cannot simply put together the location histories of users from different cities into a user-location matrix, which is neither efficient nor scalable. Second, performing collaborative inference in each city separately cannot cope with the new city problem demonstrated in Fig. 13.10a very well, as a user usually has not enough location history in a city that is new to her/him.

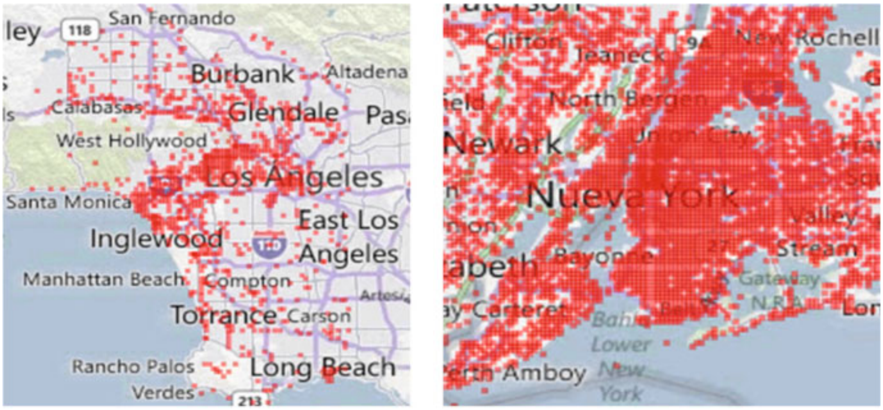


Fig. 13.10 User location distribution. (**a**) New York users in Los Angeles. (**b**) New York users in New York City

A family of location-based and preference-aware recommender system has been invented to go beyond traditional collaborative filtering recommenders to offer a particular user a set of venues (such as restaurants and shopping malls) within a user-specified geospatial range with the consideration of the three factors discussed earlier. By modeling a user's preferences based on the category information of her location history (instead of physical locations) in an Location-Based Social Network, the system can facilitate people's travel not only near their living areas but also to a city that is new to them. To this end, the system may consist of the following two phases:

Offline Modeling The offline modeling part is comprised of two major components: (1) social knowledge learning and (2) personal preference discovery. The first component infers each user's expertise in each category city by city according to their location histories. Given a predefined category hierarchy, the system breaks a user's location history in a city into groups of different location categories. Then, the system models each category group of location histories using a user location matrix, in which each entry denotes a user's number of visits to a physical location. By applying an iterative inference model to each user-location matrix, the system calculates a score w.r.t. a category for each user, indicating a user's expertise in that category in that city. By ranking the users in terms of the score corresponding to a category, we can discover the local experts of different categories in the city. The inferred expertise of a user will be used in later preference-aware candidate selection algorithm and help the online part generate quality recommendations with fewer computational loads. The second component models each user's personal preferences using a weighted category hierarchy (WCH) by taking advantage of the location category information lying her location history, which helps us to overcome the data sparsity problem. Specifically, a WCH is a sub-tree of the predefined category hierarchy, where each node carries a value denoting the user's number of visits to a category. These values are further normalized on each layer of a WCH using TF-IDF (term frequency-inverse document frequency).

Online Recommendation Generation The online recommendation part provides a user with a list of venues, considering the user's preferences, current location, and social opinions from the selected local experts, detailed in the following two components:

1. Preference-aware candidate selection. This component selects a set of local experts who visited the venues within a user's recommendation range R and has a high expertise in the categories preferred by the user. A preference-aware candidate selection algorithm is designed to properly choose these local experts from different categories according to a user's different preference weights in her/his WCH. Meanwhile, this algorithm improves the efficiency of our approach significantly while maintaining the effectiveness, making our system really location-aware.
2. Location rating calculation. This component first computes the similarity between each selected local expert and the user using a similarity function based on their WCHs. The calculated similarity score is further fed into a CF-based model to infer the rating that the user would give to an unvisited candidate venue. Later, the venues with relatively high predicted ratings are returned as the location recommendations.

13.3 Overview of Context and Preference-Aware Systems

Context-aware mobile applications need to employ a database system that can efficiently process context and preference-aware queries. Much work has gone into embedding the notion of preference and context in database systems from both the *modeling* and *implementation* aspects. The *modeling* aspect is concerned more with the theoretical foundation of preference expressions over relational data [4, 20, 21, 38, 41, 44]. In some cases, the model provides rules that define how the model translates into traditional SQL queries. For example, query personalization [41–43] models preferences using a relational graph, where preferred attributes and relations are given a degree of interest score. Using this graph, SQL queries are injected with the top-k preferences derived from the graph. Contextual databases [75, 76] focus on modeling contextual preferences and integrating context into query definitions.

On the system and implementation side, PreferenceSQL [38–40] is an extension to standard SQL supporting the best-matches only (BMO) query model. PreferenceSQL supports a number of preference operators including Pareto, Prioritization, Rank, and Dual, which can be combined to support both qualitative and quantitative preferences in a single query. The system is implemented as a middleware layer for easy integration with most database systems and supports query optimization (preference algebraic transformations, cost-based selection) as well as high-level preferences on spatial objects [83]. The PREFER system [33] incorporates preferences into a single weighted ranking function where preferred results are generated by finding precomputed materialized views whose weight function is similar to the query. AmbientDB [28, 81] provides a middleware layer to integrate myriad multimedia servers with mobile devices in order to provide efficient ad hoc queries in a dynamic mobile environment. PrefDB [8, 9] is a system for preference-aware computing that attempts to push the notion of preference closer to the DBMS. In this vein, PrefDB extends the relational data model with preference attributes and develops a preference algebra to capture the notion of preference query processing.

Context-aware database (*CareDB*) is a complete database system that (a) provides a full-fledged realization of preference and context-aware databases, (b) addresses processing preferences and context at the query operator levels, (c) exploits a built-in approach where the preference and context awareness are embedded into the core processing of query operators, and (d) is equipped with the necessary modules that support the special characteristics of location-based servers, e.g., continuous queries and dynamic environments.

Another work [8] explores embedding preferences in a relational database by extending a relation with *score* and *confidence* attributes (called *p*-relations). This framework defines a query processing operator that evaluates preferences according to the *p*-relational model and also extends traditional operators (select, project) with *p*-relation semantics. Other work has explored modeling *contextual* preferences, where the objective is to evaluate preferences that change based on a user's situation [5, 75, 76].

13.3.1 Context and Preference-Aware Database Operations

This section provides an overview of CareDB (as a case study), a full-fledged context, and preference-aware database system. The goal of CareDB is to enable the practical realization of location-based services that embed various forms of preferences and context into the core processing of location-based queries. CareDB is a complete database system—implemented in PostgreSQL—that addresses the following research challenges: (1) defining a taxonomy of preference and context types; (2) generically supporting various preference evaluation methods at all levels of the query processor, including *core* query operators (e.g., join); and (3) integrating surrounding contextual data (e.g., current traffic, weather) in core preference query processing. Contextual data calls for retrieving some attributes from computationally intense sources (e.g., third parties).

13.3.1.1 A Context-Aware Database (CareDB) System Architecture

Figure 13.11 gives an overview of the CareDB architecture. CareDB is implemented inside the PostgreSQL[8] database engine.

[8] http://www.postgresql.org

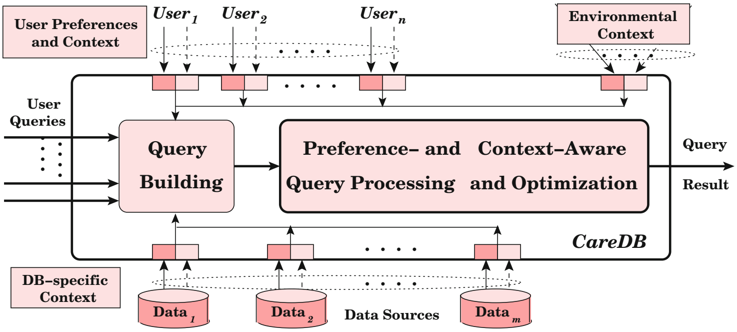


Fig. 13.11 Context and preference-aware database (*CareDB*) system architecture

CareDB Input Besides queries, CareDB takes preferences and contextual data as input. *Preferences* are a set of user's tastes for data attributes in a particular domain. For example, whenever a user searches for a restaurant, her/his profile may store preferences for minimizing travel time and price, maximizing rating, and any constraints on dietary needs. User preferences are stored in a *preference profile* and can be given to CareDB explicitly or learned through the user's history [62]. CareDB has three input *context* types: *user context*, *database-specific context*, and *environmental context*. Each context can be either *static* (rarely changed) or *dynamic* (frequently changing). Static/dynamic context is depicted by solid/dotted lines and dark/light gray rectangles, respectively, in Fig. 13.11. *User context* is any extra information about a user. Static user context data can include income, profession, and age, while dynamic attributes include current user location or status (e.g., "at home," "in meeting"). *Database context* refers to application-specific data sources (e.g., restaurant, hotel, and taxi databases) that are registered with CareDB. As an example, for a restaurant database, static context data includes price, rating, and operating hours, while dynamic context includes current waiting time. *Environmental context* is any information about surrounding environment. This data is assumed to be stored at a *third party* and accessed by the query processor during query runtime by calling the data source through a remote API (e.g., Web service interface). A dynamic environmental context includes traffic and travel time (e.g., from Yahoo Traffic[9]), while a relatively static context includes weather information (e.g., from NWS[10]).

Query Building The purpose of the query building module (rounded square in Fig. 13.11) is to *personalize* queries for each user such that the *best* answers are returned. The user submits simple queries without constraints (e.g., "Find me a restaurant"). The query building module creates *preference queries* by augmenting the submitted query with the preferences stored in the user's preference profile. The description of *preference queries* is provided next.

[9] Yahoo Traffic Web Services. http://developer.yahoo.com/traffic/

[10] National Weather Service Web Service. http://www.weather.gov/xml/

Query Processor The novelty of CareDB lies within the *preference and context-aware query processing module*. The query processor takes as input *preference queries*. To execute preference queries efficiently, CareDB uses the *FlexPref* [45] query execution framework that:

1. Extends SQL syntax with a `Preferring` clause for specifying preference objectives (e.g., minimize price, maximize distance) and a `Using` clause that specifies *which* preference method should evaluate the objectives (e.g., skyline [12], top-*k* dominance [88], *k*-dominance [17]). Examples of this syntax are given in Fig. 13.13.
2. Employs a set of generic, extensible operators *inside* the DBMS query processor to execute the preference query. In addition, CareDB provides a generic, extensible platform for preference evaluation with (a) efficient join operations, (b) expensive attributes, and (c) uncertain data.

13.3.1.2 Preference-Aware Query Processing and Optimization (FlexPref)

At the core of CareDB is the FlexPref query processing framework. FlexPref is implemented inside the PostgreSQLquery processor and is extensible to arbitrary preference methods. FlexPref consists of a set of generic relational operators that implement the query processing steps common to many preference methods. The generic operators themselves do not evaluate preference semantics; these semantics are injected into the FlexPref operators through the implementation of only three external callback functions that are then registered with FlexPref. The major advantages of FlexPref are:

1. *Ease of implementation*. Placing a preference function in FlexPref (and hence within the DBMS) requires the implementation of only three external callback functions (outside PostgreSQL) that define the semantics of the preference method and that are then registered with FlexPref.
2. *Small footprint*. FlexPref requires orders of magnitude less code to implement a preference method compared to the built-in approach. For instance, a custom skyline implementation in PostreSQLrequires 2000 lines of code, while in FlexPref it requires only 300 lines.
3. *Seamless DBMS integration*. Since FlexPref is extensible, a preference method, once registered, is instantly ready for use in arbitrary select-project-join queries.
4. *Efficiency*. FlexPref is tightly coupled with the DBMS query processor, meaning that a registered preference method realizes efficient query processing comparable to the *built-in* implementation approach.

General Functions

FlexPref requires the definition of two macros and three functions to register a new preference method. The two required macros are:

- **`#define DefaultScore`**: each object in FlexPref is associated with a score, internal to the underlying preference method. Defining a default score ensures that each object is assigned a value.
- **`#define IsTransitive`**: indicates whether the preference method exhibits transitivity or not. Knowledge of transitivity leads to better query processing efficiency.

The three general functions requiring implementation are:

- **`PairwiseCompare(Object P, Object Q)`**: Given two data objects *P* and *Q*, *update* the score of *P* and return 1 if *Q* can *never* be a preferred object, −1 if *P* can *never* be a preferred object, and 0 otherwise.
- **`IsPreferredObject(Object P, PreferenceSet S)`**: Given a data object *P* and a set of preferred objects *S*, return *true* if *P* is a preferred object and can be added to *S*, and *false* otherwise.
- **`AddPreferredToSet(Object P, PreferenceSet S)`**: Given a data object *P* and a preference set *S*, add *P* to *S* and remove or rearrange objects from *S*, if necessary.

These functions break down preference evaluation into a set of modular operations that need *not* be aware of query processor specifics.

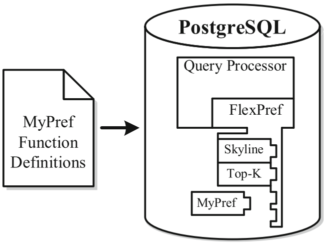
Preference Method Registration

Once the macros and external functions are implemented, a preference method is registered with FlexPref using a `DefinePreference` command:

```
DefinePreference [Name] WITH [File]
```

The *name* argument is the name of the preference method, while the *file* argument specifies the file containing the function definitions. `DefinePreference` compiles the preference code into our framework. This process is depicted in Fig. 13.12 for a preference method "MyPref."

Fig. 13.12 Flexpref



Queries in FlexPref

Once a preference method is registered with FlexPref, it can be used in database queries immediately. FlexPref requires the extension of the SQL syntax in order to select the appropriate preference methods and specify their objectives. FlexPref adds a `Preferring` and `Using` clause to conventional SQL in order to issue preference queries. A typical query in FlexPref is:

Listing 13.1 Structure of a typical query in FlexPref

```
SELECT  [Select Clause] FROM [Tables]
WHERE [Where Clause]
PREFERRING [Preference Attributes]
USING  [method] WITH [Parameter]
OBJECTIVES [Objective]
```

Here, the method (with objectives) specified in the `Using` clause is responsible for selecting the preference evaluation method to be applied over the attributes given in the `Preferring` clause. As an example, consider the following query for the well-known skyline [12] preference method implemented in FlexPref:

Listing 13.2 Example of query in FlexPref

```
SELECT  * FROM Restaurant R
PREFERRING R.price d1 AND R.dist d2 AND R.rating d3
USING Skyline WITH OBJECTIVES MIN d1, MIN d2, MAX d3;
```

This query will evaluate the skyline of restaurant data, where the preference objectives require minimizing both price and distance attributes while maximizing rating. As another example, consider the following abbreviated query for the top-*k* dominating [88] preference method:

Listing 13.3 Structure of a typical query in FlexPref

```
USING Top-K-Domination WITH K=2
OBJECTIVES MIN d1, MIN d2, MAX d3;
```

Here, the `Using` clause specifies that (1) $K = 2$ answers are required and (2) preference is based on minimizing both price and distance while maximizing rating.

FlexPref Functionality

To provide efficient query processing support for the registered preference methods, FlexPref provides extensible query operators, implemented inside the engine of the DBMS. Note that these operators require implementation in the query processor *only once*. Once in place, these operators use the general macro and function definitions to evaluate the preference method from within the query processor, requiring no further changes to the DBMS engine. Below is a brief overview of three core FlexPref extensible operators (details in [45]).

Selection The FlexPref selection algorithm evaluates the set of preferred objects from a *single table*. The main idea is to compare tuples pairwise while incrementally building a preferred answer set. During execution, a data object P may be found to be dominated (i.e., guaranteed *never* to be a preferred answer). If the underlying preference method is *transitive*, P is immediately discarded and not processed further, thus leading to more efficient execution.

Join The FlexPref join algorithm enables efficient preference evaluation for data that exists in *multiple* tables. The main idea behind this join operation involves using the general functions to *prune* tuples from the join input that are guaranteed not to be in the final answer. Pruning enhances join performance for two reasons: (a) the amount of data to be joined from input tables is greatly reduced due to pruning the input data, and (b) the amount of data processed by the final preference evaluation after the join is reduced based on the multiplying factor of the join.

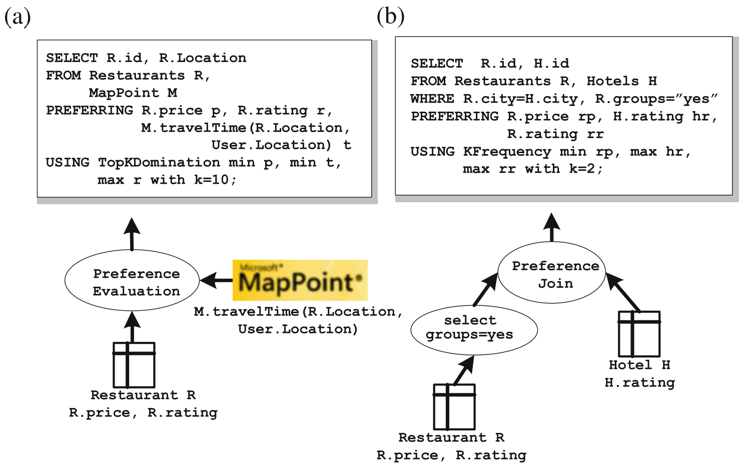
Sorted List (Index) Access The availability of sorted attributes (e.g., indexes such as the B+-tree) allows for efficient preference evaluation. The idea is that complete preference answer generation can be guaranteed after reading only a *portion* of the sorted data, thus reducing the I/O overhead compared to query processing over unsorted or non-indexed data. FlexPref exploits this idea by employing an algorithm capable of processing sorted attributes in round-robin fashion and stopping I/O once a stopping condition, provided by the general functions, has been met.

13.3.1.3 CareDB Technical Features

All *CareDB* features discussed in this section are built in a *generic* and *extensible* manner, capable of supporting multiple preference methods. The basic idea is to create a single, generic operator for each feature (e.g., uncertainty, join) that performs computations common across preference methods. Each operator is *extensible* through "plug-in" functions that tell the operator about the semantics of a specific preference method.[11] From a system's perspective, this is a powerful method for implementing preference query processing in a database, as the engine need only be chanced *once*, while existing and future preference methods can "plug into" the existing framework.

Query Processing with Expensive Attributes In *CareDB*, it is assumed that some attribute values will be expensive to derive, as the derived value may require extensive computations (e.g., road network travel time), or must be retrieved from a

[11] We refer to [45] for details of the concept of extensible preference query processing in a DBMS engine and initial work on simple query operations.



```
SELECT R.id, R.Location
FROM Restaurants R,
     MapPoint M
PREFERRING R.price p, R.rating r,
           M.travelTime(R.Location,
                   User.Location) t
USING TopKDomination min p, min t,
      max r with k=10;
```

```
SELECT  R.id, H.id
FROM Restaurants R, Hotels H
WHERE R.city=H.city, R.groups="yes"
PREFERRING R.price rp, H.rating hr,
          R.rating rr
USING KFrequency min rp, max hr,
      max rr with k=2;
```

Fig. 13.13 Preference and context-aware query examples. (**a**) Expensive attribute query. (**b**) Join query

third party (e.g., remote Web service). Figure 13.13a gives an example query (and plan) to find a preferred restaurant using the top-*k* domination method [88], where attributes *price* and *rating* are stored in a local relation, while the *travel time* attribute is requested from the Microsoft MapPoint[12] Web service based on the restaurant and user locations. Under these circumstances, computational overhead is dominated by deriving the expensive *travel time* attribute; thus, the preference query processing operator should avoid computing these expensive attributes whenever possible.

The *CareDB* query processor is designed to take these challenges into account. *CareDB* employs a preference evaluation operator that computes the preference answer by retrieving as few expensive data attributes as possible. The main idea is to first perform preference evaluation over *local* data attributes, forming a local answer set *LA* using "plug-in" functions to determine the semantics of the specific preference method used to execute the query (e.g., top-*k* domination). The operator then selectively requests expensive attributes for objects in *LA* guaranteed to be preference answers and *prunes* objects that are guaranteed not to be preference answers. *CareDB* then makes a minimum number of expensive attribute requests necessary to completely and correctly execute the preference query.

Generic Preference Join Like most query processing scenarios for real-life applications, it is likely that most data stored locally in the *CareDB* will not reside in a single table. For example, Fig. 13.13b shows an example preference query (and plan) using the *k*-dominance preference method [17] for a user requesting a hotel and restaurant pair. The preference objectives are to minimize the restaurant price and maximize both restaurant and hotel ratings. The hotel and restaurant data

[12] http://www.microsoft.com/mappoint/

is stored in separate tables, necessitating a join to answer the preference query. The naive method to answer this query is to perform the join and then perform preference evaluation. Current state-of-the-art join methods address a specific preference method (e.g., skyline join [36]), often assuming a specific index for progressive result generation [36, 61].

CareDB, on the other hand, employs *PrefJoin* [37], an efficient preference join operation that is *generic* for a wide variety of preference functions and does not assume the existence of any index structure. The goal of *PrefJoin* is to make the join operation aware of the required preference functionality through the "plug-in" functions, and hence the join operation would be able to prune those tuples early that have no chance of being a preferred object without actually doing the join operation. The PrefJoin algorithm consists of four phases, namely, *Local Pruning*, *Data Preparation*, *Joining*, and *Refining*. The *Local Pruning* phase filters out, from each input relation, those tuples that are guaranteed not to be in the final preference set. The *Data Preparation phase* associates metadata with each non-filtered tuple that will be used to optimize the execution of the next phase. The *Joining* phase uses that metadata, computed in the previous phase, to decide on which tuples should be joined together. Finally, the *Refining* phase finds the final preference set from the output of the joining phase.

References

1. Abdelhaq, H., Sengstock, C., Gertz, M.: EvenTweet: online localized event detection from Twitter. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (2013)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
3. After Boston Explosions, People Rush to Twitter for Breaking News. http://www.latimes.com/business/technology/la-fi-tn-after-boston-explosions-people-rush-to-twitter-for-breaking-news-20130415,0,3729783.story (2013)
4. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2000)
5. Agrawal, R., Rantzau, R., Terzi, E.: Context-sensitive ranking. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2006)
6. Alsubaiee, S., Altowim, Y., Altwaijry, H., Behm, A., Borkar, V.R., Bu, Y., Carey, M.J., Grover, R., Heilbron, Z., Kim, Y.S., Li, C., Onose, N., Pirzadeh, P., Vernica, R., Wen, J.: ASTERIX: an open source system for "Big Data" management and analysis. Proc. Int. Conf. Very Large Data Bases **5**(12), 1898–1901 (2012)
7. Apple buys social media analytics firm Topsy Labs. http://www.bbc.co.uk/news/business-25195534 (2013)
8. Arvanitis, A., Koutrika, G.: Towards preference-aware relational databases. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 426–437 (2012)
9. Arvanitis, A., Koutrika, G.: Prefdb: supporting preferences as first-class citizens in relational databases. IEEE Trans. Knowl. Data Eng. **26**(6), 1430–1446 (2014)
10. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press/Addison-Wesley, New York (1999)

11. Bao, J., Mokbel, M.F., Chow, C.Y.: GeoFeed: a location-aware news feed system. In: ICDE, pp. 54–65 (2012)
12. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
13. Brown, M.G.: Supporting user mobility. In: IFIP World Conference on Mobile Communications (1996)
14. Brown, P.J., Bovey, J.D., Chen, X.: Context-aware applications: from the laboratory to the marketplace. IEEE Pers. Commun. **4**(5), 58–64 (1997)
15. Budak, C., Georgiou, T., Agrawal, D., Abbadi, A.E.: GeoScope: online detection of geo-correlated information trends in social networks. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (2014)
16. Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., Lin, J.: Earlybird: real-time search at Twitter. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE (2012)
17. Chan, C.Y., Jagadish, H., Tan, K.L., Tung, A.K., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2006)
18. Chen, C., Li, F., Ooi, B.C., Wu, S.: TI: an efficient indexing mechanism for real-time search on tweets. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2011)
19. Cheng, R., Xia, Y., Prabhakar, S., Shah, R.: Change tolerant indexing for constantly evolving data. In: ICDE (2005)
20. Chomicki, J.: Querying with intrinsic preferences. In: Proceedings of the International Conference on Extending Database Technology, EDBT (2002)
21. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. **28**(4), 427–466 (2003)
22. Chow, C.Y., Bao, J., Mokbel, M.F.: Towards location-based social networking services. In: The 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks (2010)
23. Cooperstock, J.R., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W.: Evolution of a reactive environment. In: Proceedings of the International Conference on Human Factors in Computing Systems, CHI (1995)
24. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
25. Dey, A.K., Abowd, G.D., Wood, A.: CyberDesk: a framework for providing self-integrating context-aware services. Knowl.-Based Syst. **11**(1), 3–13 (1998)
26. Elrod, S., Hall, G., Costanza, R., Dixon, M., des Rivières, J.: Responsive office environments. Commun. ACM **36**(7), 84–85 (1993)
27. Facebook Statistics. https://www.facebook.com/business/power-of-advertising (2012)
28. Feng, L., Apers, P.M.G., Jonker, W.: Towards context-aware data management for ambient intelligence. In: International Conference of Database and Expert Systems (2004)
29. Fickas, S., Kortuem, G., Segall, Z.: Software organization for dynamic and adaptable wearable systems. In: International Symposium on Wearable Computers, pp. 56–63 (1997)
30. Güting, R.H., de Almeida, V.T., Ansorge, D., Behr, T., Ding, Z., Höse, T., Hoffmann, F., Spiekermann, M., Telle, U.: SECONDO: an extensible DBMS platform for research prototyping and teaching. In: Proceedings of the International Conference on Data Engineering, ICDE (2005)
31. Harvard Tweet Map. http://worldmap.harvard.edu/tweetmap/ (2013)
32. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsioutsiouliklis, K.: Discovering geographical topics in the twitter stream. In: Proceedings of the International Conference on World Wide Web, WWW (2012)
33. Hristidis, V., Koudas, N., Papakonstantinou, Y.: PREFER: a system for the efficient execution of multi-parametric ranked queries. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2001)
34. Hull, R., Neaves, P., Bedford-Roberts, J.: Towards situated computing. In: International Symposium on Wearable Computers (1997)

35. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S.: Path prediction and predictive range querying in road network databases. VLDB J. **19**(4), 585–602 (2010)
36. Jin, W., Morse, M., Patel, J., Ester, M., Hu, Z.: Evaluating skylines in the presence of equijoins. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 249–260 (2010)
37. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Prefjoin: an efficient preference-aware join operator. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 995–1006 (2011)
38. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (2002)
39. Kießling, W., Köstler, G.: Preference SQL: design, implementation, experiences. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (2002)
40. Kießling, W., Endres, M., Wenzel, F.: The preference sql system - an overview. IEEE Data Eng. Bull. **34**(2), 11–18 (2011)
41. Koutrika, G., Ioannidis, Y.: Personalization of queries in database systems. In: Proceedings of the International Conference on Data Engineering, ICDE (2004)
42. Koutrika, G., Ioannidis, Y.: Constrained optimalities in query personalization. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (2005)
43. Koutrika, G., Ioannidis, Y.E.: Personalized queries under a generalized preference model. In: Proceedings of the International Conference on Data Engineering, ICDE (2005)
44. Lacroix, M., Lavency, P.: Preferences: putting more knowledge into queries. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (1987)
45. Levandoski, J.J., Khalefa, M., Mokbel, M.F.: FlexPref: a framework for extensible preference evaluation in database systems. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 828–839 (2010)
46. Levandoski, J., Sarwat, M., Eldawy, A., Mokbel, M.: LARS: a location-aware recommender system. In: ICDE, pp. 450–461 (2012)
47. Levandoski, J.J., Sarwat, M., Mokbel, M.F., Ekstrand, M.D.: RecStore: an extensible and adaptive framework for online recommender queries inside the database engine. In: Proceedings of the International Conference on Extending Database Technology, EDBT (2012)
48. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.C.: TEDAS: a Twitter-based event detection and analysis system. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE (2012)
49. Linden, G., et al.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
50. Liu, W., Zheng, Y., Chawla, S., Yuan, J., Xing, X.: Discovering spatio-temporal causal interactions in traffic data streams. In: Proceedings of the ACM International Conference on Knowledge and Data Discovery, KDD (2011)
51. Magdy, A., Alarabi, L., Al-Harthi, S., Musleh, M., Ghanem, T., Ghani, S., Mokbel, M.: Taghreed: a system for querying, analyzing, and visualizing geotagged microblogs. In: Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM GIS (2014)
52. Magdy, A., Aly, A.M., Mokbel, M.F., Elnikety, S., He, Y., Nath, S.: Mars: real-time spatio-temporal queries on microblogs. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE, pp. 1238–1241 (2014)
53. Magdy, A., Mokbel, M.F., Elnikety, S., Nath, S., He, Y.: Mercury: a memory-constrained spatio-temporal real-time search on microblogs. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE, pp. 172–183 (2014)
54. Marcus, A., Bernstein, M.S., Badar, O., Karger, D.R., Madden, S., Miller, R.C.: Twitinfo: aggregating and visualizing microblogs for event exploration. In: Proceedings of the International Conference on Human Factors in Computing Systems, CHI (2011)
55. Mokbel, M.F., Aref, W.G.: PLACE: a scalable location-aware database server for spatio-temporal data streams. IEEE Data Eng. Bull. **28**(3), 3–10 (2005)

56. Mokbel, M.F., Xiong, X., Aref, W.G.: SINA: scalable incremental processing of continuous queries in spatio-temporal databases. In: SIGMOD (2004)
57. Mokbel, M.F., Xiong, X., Aref, W.G., Hambrusch, S., Prabhakar, S., Hammad, M.: PLACE: a query processor for handling real-time spatio-temporal data streams (Demo). In: Proceedings of the International Conference on Very Large Data Bases, VLDB (2004)
58. New Features on Twitter for Windows Phone 3.0. https://blog.twitter.com/2013/new-features-on-twitter-for-windows-phone-30 (2013)
59. Phelan, O., McCarthy, K., Smyth, B.: Using twitter to recommend real-time topical news. In: Proceedings of the ACM Conference on Recommender Systems, RecSys (2009)
60. Prabhakar, S., Xia, Y., Kalashnikov, D.V., Aref, W.G., Hambrusch, S.E.: Query indexing and velocity constrained indexing: scalable techniques for continuous queries on moving objects. IEEE Trans. Comput. **51**(10), 1124–1140 (2002)
61. Raghavan, V., Rundensteiner, E.: Progressive result generation for multi-criteria decision support queries. In: Proceedings of the International Conference on Data Engineering, ICDE, pp. 733–744 (2010)
62. Rashid, A.M., Albert, I., Coslely, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the International Conference on Intelligent User Interfaces (2002)
63. Rekimoto, J., Ayatsuka, Y., Hayashi, K.: Augment-able reality: situated communication through physical and digital spaces. In: International Symposium on Wearable Computers (1998)
64. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: CSWC (1994)
65. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: Proceedings of the International Conference on World Wide Web, WWW (2010)
66. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D., Sperling, J.: TwitterStand: news in tweets. In: Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM GIS (2009)
67. Sarwat, M., Bao, J., Eldawy, A., Levandoski, J.J., Magdy, A., Mokbel, M.F.: Sindbad: a location-based social networking system. In: SIGMOD, pp. 649–652 (2012)
68. Sarwat, M., Avery, J., Mokbel, M.F.: RecDB in action: recommendation made easy in relational databases. PVLDB **6**(12), 1242–1245 (2013)
69. Sarwat, M., Levandoski, J.J., Eldawy, A., Mokbel, M.F.: LARS*: an efficient and scalable location-aware recommender system. IEEE Trans. Knowl. Data Eng. **26**(6), 1384–1399 (2014)
70. Schilit, B.N., Adams, N.I., Want, R.: Context-aware computing applications. In: Workshop on Mobile Computing Systems and Applications (1994)
71. Silberstein, A., Terrace, J., Cooper, B.F., Ramakrishnan, R.: Feeding frenzy: selectively materializing user's event feed. In: Proceedings of the ACM International Conference on Management of Data, SIGMOD, pp. 831–842 (2010)
72. Sina Weibo, China's Twitter, comes to rescue amid flooding in Beijing. http://thenextweb.com/asia/2012/07/23/sina-weibo-chinas-twitter-comes-to-rescue-amid-flooding-in-beijing/ (2012)
73. Singh, V.K., Gao, M., Jain, R.: Situation detection and control using spatio-temporal analysis of microblogs. In: Proceedings of the International Conference on World Wide Web, WWW (2010)
74. Skovsgaard, A., Sidlauskas, D., Jensen, C.S.: Scalable top-k spatio-temporal term querying. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE, pp. 148–159 (2014)
75. Stefanidis, K., Pitoura, E.: Fast contextual preference scoring of database tuples. In: Proceedings of the International Conference on Extending Database Technology, EDBT (2008)
76. Stefanidis, K., Pitoura, E., Vassiliadis, P.: Adding context to preferences. In: Proceedings of the International Conference on Data Engineering, ICDE (2007)
77. Topsy Pro Analytics: Find the insights that matter. http://topsy.com/ (2013)

78. TweetTracker: track, analyze, and understand activity on Twitter. http://tweettracker.fulton.asu.edu/ (2013)
79. Twitter Data Grants.. https://blog.twitter.com/2014/introducing-twitter-data-grants (2014)
80. Twitter Statistics. http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/ (2013)
81. van Bunningen, A.H., Feng, L., Apers, P.M.G.: A context-aware preference model for database querying in an ambient intelligent environment. In: International Conference of Database and Expert Systems (2006)
82. Watanabe, K., Ochi, M., Okabe, M., Onai, R.: Jasmine: A real-time local-event detection system based on geolocation information propagated to microblogs. In: Proceedings of the ACM International Conference on Information and Knowledge Management, CIKM (2011)
83. Wenzel, F., Endres, M., Mandl, S., Kießling, W.: Complex preference queries supporting spatial applications for user groups. Proc VLDB Endowment **5**(12), 1946–1949 (2012)
84. Wolfson, O., Sistla, A.P., Xu, B., Zhou, J., Chamberlain, S.: DOMINO: databases for MovINg objects tracking (Demo). In: Proceedings of the ACM International Conference on Management of Data, SIGMOD (1999)
85. Wu, L., Lin, W., Xiao, X., Xu, Y.: LSII: an indexing structure for exact real-time search on microblogs. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE (2013)
86. Xu, W., Chow, C.Y., Yiu, M.L., Li, Q., Poon, C.K.: MobiFeed: a location-aware news feed system for mobile users. In: SIGSPATIAL (2012)
87. Yao, J., Cui, B., Xue, Z., Liu, Q.: Provenance-based indexing support in micro-blog platforms. In: Proceedings of the IEEE International Conference on Data Engineering, ICDE (2012)
88. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data. In: VLDB, pp. 483–494 (2007)

Part V
Multimedia Information Management

The research challenge addressed in this part is the effective and efficient management of multimedia information (e.g., video, audio, images, texts, three-dimensional (3D) models, etc.) within pervasive environments, especially for cultural heritage applications. In particular, the main current research issues are addressed, which aim at providing novel techniques for supporting query by content and example mechanisms for multimedia databases and digital libraries, multimedia recommendation, and media content presentation strategies, multimedia applications, and services.

Generally, content-based multimedia information retrieval provides new models and methods for effectively and efficiently searching through the huge variety of media that are available in different kinds of repositories (digital libraries, Web portals, social networks, multimedia databases, etc.).

Chapter 14 provides the foundations and the current state of the art of content-based multimedia information retrieval, including the most promising browsing and search paradigms for the several types of multimedia data, analyzing their role in cultural heritage applications. Eventually, the authors discuss the major challenges in future researches.

The intrinsic complexity and diversity of data in multimedia digital libraries (MDLs) require devising techniques and solutions that are inherently different from those usually adopted in traditional information retrieval and database systems. Moreover, the size and the dynamicity of MDLs force researchers to strive for efficiency, so as to guarantee real-time results to the users. Finally, semantics should also be brought into play in order to facilitate users' experience in querying, browsing, and consuming multimedia information.

Chapter 15 presents an approach toward the efficient, effective, and semantically rich data retrieval in MDLs. With respect to the commonly used holistic approach, where the multimedia datum is considered as an atomic entity, our reductionist strategy considers the multimedia information as a complex combination of component subparts and eases the fulfillment of the three above properties of efficiency, effectiveness, and semantic richness. Indeed, by decomposing multimedia information into simpler and smaller component objects, we are able to index such components without giving up the ability to query the original information as a whole.

As a further issue, in the last decade the spread of broadband Internet connections even for mobile devices has contributed to an increased availability of multimedia information on the Web. At the same time, due to the decrease of storage cost and the increasing popularity of storage services in the cloud, the problem of information overload has become extremely serious even in personal/company archives. The need of managing, retrieving, and presenting all these data has promoted the development of advanced multimedia information systems based on models and techniques proposed in the scientific literature. Nowadays, detecting frameworks that account for the specificity of target domains, such as the cultural heritage domain, is an important challenge. This context is really interesting both for its impact on the users' life and education and also because it requires to rethink the existing models for many different aspects, including (a) the different roles of each heterogeneous data, (b) how the importance of users' patterns can be extracted, as well as (c) how data have to be presented in an integrated way. Most of these challenges have been extensively studied in the area of multimedia recommendation techniques, in which multiple representations of the same information need to be handled. A video, for instance, can be described by a temporal sequence of frames; a textual description; a set of users' tags, comments, or emotions; a set of users that watched it; a set of similar videos; and so on. However, these features are not equally important for everyone. For example, in the choice of the next video to watch, a user may be guided by descriptions, emotions, tags, users' communities, or even combinations of these descriptors. A good multimedia recommender system should address the specificity of each user in browsing and choosing the multimedia content.

In this scenario, Chap. 16 investigates how multimedia information systems in the cultural heritage domain can significantly benefit from the application of properly tuned techniques borrowed by state-of-the-art multimedia recommendation and delivery systems. To support their claim, the authors propose a survey of such existing techniques and discuss their potential impact in the design of multimedia information system in the cultural heritage domain.

Chapter 14
Content-Based Multimedia Retrieval

Flora Amato, Luca Greco, Fabio Persia, Silvestro Roberto Poccia, and Aniello De Santo

14.1 Introduction

During the last years, rapid diffusion of inexpensive tools (mostly embedded in smartphones, tablets, and other common mobile devices) for image, video, and audio capturing, together with improvements in storage technologies, have favored the creation of massive collections of digital multimedia content. However, the difficulty of finding relevant items increases with the growth of the amount of available data. To face this problem, two main approaches are often employed, both involving the use of metadata: the first includes manually annotating multimedia content by means of textual information (e.g., titles, descriptive keywords from a limited vocabulary, and predetermined classification schemes), while the second amounts to using automated feature extraction and object recognition to classify content.

The latter is naturally related to *content-based multimedia retrieval* (CBMR) systems and often represents the only viable solution since text annotations are mostly nonexistent or incomplete; moreover, CBMR has proven to dramatically reduce time and effort to obtain multimedia information, whereas frequent annotation additions and updates to massive data stores often require entering manually all the attributes that might be needed for the queries.

While early algorithms proposed in this field mainly focused on feature-based similarity search (over images, video, and audio), the interest of researchers

F. Amato (✉) • F. Persia • S.R. Poccia • A. De Santo
DIETI, University of Naples Federico II, Naples, Italy
e-mail: flora.amato@unina.it; fabio.persia@unina.it; silvestroroberto.poccia@unina.it; aniello.desanto@gmail.com

L. Greco
DIEM, University of Salerno, Fisciano, Italy
e-mail: lgreco@unisa.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_14

has recently turned to the problem of understanding the semantics of a query rather than concentrating only on the optimization of the underlying computation (*semantic gap*). In this direction, interesting research topics include various kinds of interaction between humans and computers (experiential and affective computing), the introduction of new features to improve the detection/recognition process, the analysis of new media types (3D models, virtual reality, etc.), and the identification of test sets for benchmarking.

In this chapter, we describe the most representative work in the field of content-based multimedia retrieval. In particular, Sect. 14.2 presents a quick survey on text retrieval models and methods. In Sect. 14.3, we report the main trends and approaches for images and video retrieval. In Sect. 14.4, main researches and applications in content-based audio retrieval are covered. Finally, in Sect. 14.5 some cultural heritage applications and further domains for content-based multimedia retrieval are briefly discussed.

14.2 Content-Based Text Retrieval

Already in 1961, Swanson [53] had defined the process of indexing, cataloguing, and classifying documents as an attempt to represent information content within articles in an extremely compact form. He established the two fundamental objectives for information retrieval (IR) systems:

- *Effectiveness*, which concerns how well a system works in terms of percentage of relevant material retrieved and amount of irrelevant material excluded.
- *Economy*, which refers, on the one hand, to the cost of the overall working system, including indexing, storage, and searching, and, on the other hand, to the cost for the user to read irrelevant material.

Current search engines can recognize strings of characters and figures, but, unfortunately, they do not have the key to enter the semantic dimension of words: they cannot understand the richness and the imperfections of natural language. Therefore, even though users try to avoid the muddle of links and knots of the Web, resorting to search engines for help, they still cannot take for granted that the results of their searches will be satisfactory. Often, users searching the Web by means of a Web search engine meet the following obstacles:

- *False positives*: some retrieved Web pages do not match the actual user's intention.
- *False negatives*: some pages matching the user needs are not suggested by the system.

- *Hidden Web*: some interesting pages are in the Web but are not "visible" to the crawler[1] and thus have not been indexed.

A Web search engine is actually the most common example of an IR system. One such system can assist and even influence the activity of many users: knowledge workers, intellectual workers, top managers, and, in general, users who face the problem of information overload on a daily basis [27] creating value from the chaos [38]. In detail, IR involves a specific automatic text analysis activity that focuses on large collections of documents and aims at performing searches through specific queries.

Such queries consist mostly of words, phrases, and sentences written in natural language; moreover, complex Boolean expressions, regular expressions, or finite-state automata can be used [45].

14.2.1 Basic Crawling and Indexing Strategies

Documents can be crawled using several strategies:

- *Orthographic*, where words are treated just as strings of characters
- *Semantic*, where words are connected with the concepts they express
- *Statistical*, where the term frequency is systematically compared with a frequency lexicon [11]

In general, a document can be represented as a set of keywords (or key phrases) that contribute to the description of its content. During the indexing phase, when the collection and the storage of the data are performed, texts are usually preprocessed in order to remove *stop words*[2] and to perform *stemming*.[3] Additionally, the obtained words and stems can be reconnected to their synonyms in order to create relations between words and concept classes [32].

IR systems try to retrieve all the documents that are relevant to a user query while minimizing the number of nonrelevant documents retrieved.

[1] A *Web crawler* is the component of a Web search engine that systematically browses the World Wide Web (WWW), typically for the purpose of Web indexing. A Web crawler may also be called a Web *spider* or *automatic indexer*.

[2] Stop words are words that are filtered out before processing natural language data (text), such as articles or prepositions.

[3] The process of reducing inflected or derived words to their stem or morphological root, i.e., the part of the word that is common to all its inflected variants.

14.2.2 IR Models and Weighting Schemes

Information retrieval models are classified into three different categories [58]:

- *Algebraic methods* [80], where documents are represented as vectors, matrices, or tuples that are transformed into a one-dimensional similarity measure using algebra. Examples include the vector space model [81] that is the most used model in IR.
- *Probabilistic approaches* [78], where the relevance of a document depends on the weight of the contained words.
- *Boolean strategies* [12], where documents are represented by sets of terms; the similarities are derived using set-theoretic operations, and the model simply checks the presence/absence of queried terms in the documents. Examples of these models are the standard Boolean model, the extended Boolean model, and the fuzzy model.

The standard Boolean, extended Boolean, vector space, and probabilistic techniques are also known as *statistical methodologies* [22].

Generally speaking, the statistical approaches represent documents and queries using features such as *terms* (words that occur in a given query or collection of documents) or *n-grams* (n-adjacent words) or other aspect obtained through the use of natural language processing methods [2]. Numerical weights can be assigned to each feature.

Among the weighting functions proposed in the literature [80], the following must be recalled:

- The mere frequency of occurrences of a term within a document
- The term specificity [73]
- The inverse document frequency, that is, a word is considered as less relevant if it is too frequent within a corpus of documents [80]
- The term frequency-inverse document frequency (TF-IDF) [73], whose value increases proportionally to the number of times a word appears in the document but is counterbalanced by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general

Other methods that are worth mentioning are the ones based on document-to-document similarity functions, where one can understand how close the content is of two different documents [44, 80]. These are procedures for document classification and clustering and may be *supervised*, if the classes are defined at the start of the process [21], or *unsupervised*, if a preliminary text analysis is required in order to acquire relevant homogeneous conceptual areas and classes [11].

Note that, in many applications, the results produced by information retrieval systems can be misleading: this happens because the processing of natural language is not always accurate and the semantic and syntactic aspects of language are not actually taken into account. Ontologies represent a first solution to this problem by allowing the hierarchical organization of the semantic aspects and the relations involved in the encyclopedic human knowledge. Ontology-oriented approaches to IR have been proposed by several authors [20, 35, 49, 70].

While ontologies are mostly based on nouns, a proposal that comes from the linguistics research area grounds its results on the connection between the syntactic structures and the semantics of the lexicon-grammar semantic predicates (verbs, their nominalization, and their adjectivalizations) [75]. In this field, other contributions that need to be mentioned are [23, 29, 30]. The use of syntactic relationships could increase the effectiveness by reducing the amount of irrelevant material with no loss of what is relevant. This issue has already been studied in the 1970s by Z.S. Harris and the University of Pennsylvania [43]. The use of semantics, instead, helps information retrieval to reduce irrelevant material by applying semantic techniques to disambiguate terms within documents: for instance, to distinguish between documents about *Apple* the computer company from documents about the fruit.

In general, the approaches that make use of semantic information can belong to different classes [39, 47]:

- *Neurolinguistic programming (NLP)-based methods* that ground the semantic analysis of free texts on the syntactic, lexical, and morphological ones [92]
- *Latent semantic indexing methods* that overperformed the traditional vector space techniques [52]
- *Neural nets methods*, based on the spreading activation search model (network nodes correspond to concepts contained into a thesaurus) [83]

In order to improve the performance of IR techniques, several extensions have been proposed; as an example, by embedding search concept or context into the query, or using query expansion and ranking techniques:

- *Conceptual IR* [28] that developed a concept-based retrieval method, relaying on Wikipedia-based explicit semantic analysis [37] and using hand-built thesauri with relations between words and concepts.
- *Contextual IR* [9, 87] aims at improving IR performances by integrating contextual information into the retrieval process and by adapting the results on users' needs. Users' profile and preferences can be profitably exploited to represent context and improve IR effectiveness [59, 67].
- *Automatic query expansion (AQE)*: AQE is an extremely promising technique to improve the retrieval effectiveness of document ranking, outdoing the reluctance and the difficulty of users in providing a more precise description of their information needs. In [14] and [19], the authors illustrate how AQE improves recall and precision of an IR system, describing the main computational aspects of this framework, comprising data acquisition and preprocessing, candidate feature generation and ranking, feature selection, and query reformulation.
- *Ranking and relevance feedback*: When the user gives a query, it is expected that she/he gets the documents most relevant to the query. For this reason, the obtained

documents are then ranked according to their degree of relevance, importance, etc. In addition, relevance feedback (RF) is one of the classical ways of refining search engine rankings: search engine first generates an initial set of rankings; then users select the relevant documents within this ranking, and based on the information in these documents, a more appropriate ranking is presented (e.g., the query may be expanded using the terms contained in the first set of relevant documents). In [77], the problem of ranking linked data is discussed, giving a comprehensive overview of existing ranking methods for the Web of data.

14.3 Content-Based Video and Image Retrieval

With the increasing proliferation of digital video contents, efficient techniques for analysis, indexing, and retrieval of videos according to their contents have become more and more important. A common first step for most content-based video analysis techniques available is segmenting a video into elementary shots, which can be composed to form a video sequence during video sorting or editing with either cut transitions or gradual transitions of visual effects, such as fades, dissolves, and wipes.

Shot boundaries are typically found by computing an image-based distance between adjacent frames and noting when the distance exceeds a certain threshold. The distance between adjacent frames can be based on statistical properties of pixels [48], compression algorithm [7], or edge differences [94]. The most widely used method is based on histogram differences.

In recent years, research has focused on the use of internal features of images and videos computed in an automated or semiautomated way [31]. The common strategy for automatic indexing had been based on using syntactic features alone. However, due to its complexity of operation, there has been a paradigm shift in the research of identifying semantic features [34]. User-friendly content-based retrieval (CBR) systems operating at semantic level would identify motion features as the key besides other features like color, objects, etc., as motion (either of camera motion or shot editing) adds to the meaning of the content.

Moreover, with the availability of image-capturing devices such as digital cameras and image scanners, the size of digital image collection is significantly increasing as well. Efficient image searching, browsing, and retrieval tools are required by users from many different domains. To this end, many general-purpose image retrieval systems have been developed. They can be classified in two different categories: *text based* and *content based*. In the text-based approach, the images are manually annotated by text descriptors that are then used by a DBMS to perform image retrieval. There are two main disadvantages with this approach. The first is that a considerable level of human effort is required for manual annotation. The second is the annotation inaccuracy due to the subjectivity of human perception [26, 84]. To overcome these disadvantages in text-based retrieval systems, the content-based image retrieval (CBIR) has been introduced. In CBIR, images are indexed by their visual content, such as color, texture, and shapes. A pioneering work was published by Chang in 1984 [15]. In the last years, several commercial products and experimental prototype systems have been developed, such as QBIC [33], Photobook [68], Virage [41], VisualSEEK [86], Netra [57], and SIMPLIcity [89].

The section is organized as in the following: Section 14.3.1 describes the possible categories of *video indexing and retrieval technique*, starting from the *shot segmentation* process, which is a preliminary phase essential for content-based video retrieval. Eventually, Sect. 14.3.2 shows the most important techniques for *content-based image retrieval*, first introducing the very important concept of *semantic gap* that exists between low-level and high-level features and then describing both the high-level semantic-based image retrieval and the low-level image features.

14.3.1 Content-Based Video Retrieval

As mentioned before, the shot segmentation is an essential phase for content-based video retrieval. A shot is defined as the consecutive frames from the start to the end of recording in a camera. It shows a continuous action in an image sequence [42]. There are two different types of transitions that can occur between shots: abrupt (discontinuous) also referred to as cut and gradual (continuous) such as fades, dissolves, and wipes.

There are several different approaches for detecting the shot in a video. The most commonly used are listed below:

- Shot boundary detection scheme based on rough-fuzzy set
- Shot boundary detection in low-pass filtered histogram space
- The hidden Markov model technique
- Shot change detection based on sliding window method
- Histogram-based detection
- Shot segmentation by graph partitioning
- Key frame extraction
- Key frame selection using adaptive temporal sampling
- Feature extraction
- Clustering
- Clustering algorithm based on K-L divergency
- Hierarchical clustering method

After performing one of the possible shot segmentation approaches, some *video indexing techniques* can be applied. There are two main possible categories of indexing techniques:

- Syntactic indexing
- Semantic indexing

As regards *syntactic indexing*, some of the prominent content-based retrieval (CBR) systems are IBM's QBIC [36], ViBE [17], VisualSEEK [86] and VideoQ [16], Photobook [68] and FourEyes [69] at MIT., Chabot [64], MARS [61], Virage [8], and Jacob [6]. These systems use syntactic features as the basis for matching and employ either *query by example* or *query-through-dialog box* to interface with the user. Thus, they operate at a lower level of abstraction, and therefore, the user needs to be highly versed in the details of the CBR system to take advantage of them.

On the contrary, in *semantic indexing*, a number of psychological studies and experiments emphasize the need for extracting the semantic information from images and video data. The two important researches in this direction are:

- Demonstrating that higher similarity ratings are produced by perceptually relevant semantic features as opposed to features derived from color histograms on the images [79].
- The performance and the efficiency of searching are generally greatly improved by using semantic cues [66] as compared to when low-level features are employed.

In summary, there is a great need to extract semantic indices for making the CBR system serviceable to the user. Though extracting all such indices might not be possible, there is great scope for furnishing the semantic indices with a certain well-established structure.

Video contains multiple types of audio and visual information, which are difficult to extract, combine, or trade off in general video information retrieval. It is based on the following concepts:

- Similarity measure
- Video retrieval using visual information
- Textual query for video retrieval
- Refinement and relevance feedback
- Pseudo-relevance feedback
- Negative pseudo-relevance feedback

14.3.2 Content-Based Image Retrieval

The main difference between *content-based* and *text-based* retrieval systems is that human interaction is an indispensable part of the latter system. Humans tend to use high-level features (concepts) to interpret images and measure their similarity. In general, there is no direct link between the high-level concepts and the low-level features [84]. Though many complex algorithms have been designed to describe color, shape, and texture features, these algorithms cannot adequately model image semantics and have a lot of limitations while dealing with broad content image databases [63].

In [26], Eakins mentioned three levels of queries in CBIR:

- *Level 1*: Retrieval by primitive features such as color, texture, shape, or the spatial location of image elements
- *Level 2*: Retrieval of objects of given type identified by derived features, with some degree of logical inference
- *Level 3*: Retrieval by abstract attributes, involving a big amount of high-level reasoning about the aim of the objects or scenes depicted

A CBIR system should provide full support in bridging the *semantic gap* between numerical image features and the richness of human semantics [85, 95] in order to support query by high-level concepts.

The state-of-the-art techniques in reducing the *semantic gap* include mainly five categories:

- Using object ontology to define high-level concepts
- Using machine learning tools to associate low-level features with query concepts
- Introducing relevance feedback (RF) into retrieval loop for continuous learning of users' intention
- Generating semantic template (ST) to support high-level image retrieval
- Making use of both the visual content of images and the related textual information (e.g. tags, keywords, descriptions, etc.) from the Web

Retrieval at Level 3 is difficult and less common. Possible Level 3 retrieval can be found in domain-specific areas such as art museums or newspaper libraries. Current systems mostly perform retrieval at Level 2. There are three fundamental components in these systems:

1. Low-level image feature extraction
2. Similarity measure
3. *Semantic gap* reduction

Low-level image feature extraction is the basis of CBIR systems. To perform CBIR, image features can be either extracted from the entire image or from regions (region-based image retrieval, RBIR).

To perform RBIR, the first step is to implement image segmentation. Then, low-level features such as color, texture, shape, or spatial location can be extracted from the segmented regions. Similarity between two images is defined based on region features.

14.4 Content-Based Audio Retrieval

Nowadays, content-based audio retrieval systems can be crucial for several application domains: from music retrieval to speech recognition, including audio segmentation, environmental sound recognition, and acoustic surveillance. As seen for text, images, and video retrieval problems, the identification of the right content-based features is the most important challenge in the design of a good audio retrieval system. In this particular case, the process of extracting features from audio signals is aimed at obtaining a compact and machine-processable description of the meaningful information within those signals. The most common features, such as mel-frequency cepstral coefficients (MFCCs), which were used primarily for speech recognition, are often used for other domains so that the problem of feature identification is generally addressed independently from the specific domain. That said, we can identify the following main fields of research for content-based retrieval systems:

- *Segmentation*, which aims at distinguishing different types of sound (music, silence, speech, etc.) by identifying homogeneous parts in an audio stream; once types have been identified, each one will be processed by means of a particular technique.
- *Music information retrieval*, which deals with the retrieval of similar pieces of music, artists, and genres [25]. The challenging problem of *music transcription* is also addressed although it requires a much deeper analysis to extract pitch, attack, duration, and signal source of each sound in a music track [50].
- *Automatic speech recognition* refers to the recognition of spoken word, language [74], and the extraction of emotions.
- *Environmental sound retrieval* includes types of sound that are neither speech nor music. The problem of audio surveillance is also addressed.

The identification of similar audio content, as perceived by humans, is actually an *inverse* problem. In fact, it deals with the estimation of model parameters by processing observed data. A semantic gap is then introduced because of the mismatch between high-level concepts and low-level description: a classical music track is seen (at a low level) as a series of samples (numeric values) by computers, but it is actually a sequence of notes with specific pitch and durations. Humans can bridge the semantic gap thanks to prior knowledge, so they perceive motifs, themes, and also emotions. The great challenge for the research community today is to make machines able to complete the task and narrow the semantic gap.

14.4.1 A Framework for Audio Retrieval Systems

A typical framework for content-based audio retrieval consists of three main modules [62]:

- An *input module* which aims at extracting features from one or multiple tracks contained in an *audio database*. As seen for text, video, and image retrieval problems, a feature extraction process is required to reduce the amount of data to be handled by the system. In the particular case of audio, the raw waveform would be too big for direct processing and also inadequate for retrieval. Common feature extraction techniques for audio ensure a size reduction of several orders of magnitude. Extracted features are stored in a *feature database*.

- A *query module* that allows the users to interact with the system. A user can provide a query object that contains audio fragments of interest (examples) as well as hummed or whistled melodies. Features must also be extracted from the query object to make a similarity comparison between the query and feature database items possible.
- A *retrieval module* which is the core of the system. Here the similarity between different feature-based media descriptions is computed.

Also for audio retrieval, the most used approach for similarity estimation relies on the vector space model so that a distance measurement is actually performed. Unfortunately, mathematical metrics seem not to match well human perception of similarity. This often leads to low-quality retrieval results, especially when the query itself is imperfect (humming or whistling). In most cases, the user has also the possibility to give a relevance feedback [55], by specifying which retrieved object is relevant for her/his needs. The quality of extracted features plays a very important role in the retrieval task: it is necessary to capture audio properties that show high variation across the available audio samples so that selected features carry only meaningful information and allow to imitate human perception by filtering, for example, components of the original signal which are not perceivable by humans. Typically, different content-based audio features allow to capture different information that can be used for similarity purposes; for example, *pitch* is useful for determining the musical note from an audio signal, while *mel-frequency cepstral coefficients* (MFCCs) are more suitable for the analysis of timbral characteristics.

14.4.2 Properties of Audio Signals

The first distinction we can make about audio signals is between tones and noise [62]: *tones* are "capable of exciting an auditory sensation having pitch,"[4] while noise typically has no pitch. Moreover, we can distinguish between *pure tones* where "the instantaneous sound pressure is a simple sinusoidal function of time" and *complex tones* that contain "sinusoidal component of different frequencies."[5] Complex tones are usually categorized into *harmonic* and *inharmonic* complex tones that contain partials with frequencies, respectively, at integer or non-integer multiples of the fundamental frequency. Noise can also be classified according to its temporal and spectral characteristics: for example, *stationary noise* is characterized by "negligibly small fluctuations of level within the period of observation," *broad-*

[4] ANSI/ASA S3.20-1995 (R2008) Bioacoustical Terminology at http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI%2FASA+S3.20-1995+%28R2008%29.

[5] Ibid.

band noise has no pitch, *white noise* equally contains all frequencies within a band, and *colored noise* has a spectral power function of frequency. Audio signals can also be described in terms of psychoacoustic attributes:

- *Duration* is the time between the start and the end of an audio signal and is typically divided into *attack, decay, sustain,* and *release* (these parameters describe the envelope of the sound and in some cases are not all present).
- *Loudness* is related to sound pressure level changes and is defined as “the attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from soft to loud.”
- *Pitch* is “the attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high.” It is often used as a synonym of the fundamental frequency.
- *Timbre* is the most complex attribute. It is defined as “the attribute of auditory sensation which enables a listener to judge that two nonidentical sounds, similarly presented and having the same loudness and pitch, are dissimilar.” Different musical instruments playing the same musical note actually have a different timbre.

The auditory perception of these attributes is really complex since they cannot be considered as independent. For example, sound pressure and the waveform may alter pitch perception, while loudness is influenced by sound duration (longer sounds appear louder). In order to take into account these issues, proper features have been designed. In the next section, a brief survey on recent audio feature extraction works is presented.

14.4.3 Audio Feature Extraction and Classification Researches

Audio feature sets must be designed in order to properly represent each aspect or attribute of audio signals [60, 88]. Different research works have been proposed in this area. In [93], some fundamental features are discussed: loudness, tone,[6] pitch, and mel-frequency cepstral coefficients (MFCCs); these are typically extracted from audio frames having a duration between 25 and 40 ms. The work in [62] proposes an audio feature taxonomy for content-based audio retrieval and compares properties of state-of-the-art and traditional features. Early works focus on pitch detection. In particular, [88] discusses the use of pitch histograms as a method to represent the pitch content of music signals in both symbolic and audio forms; a multiple-pitch detection algorithm for polyphonic signals is used to calculate pitch histograms for audio signals. Such a method has proven to obtain good results in automatic musical genre classification. In [51], a novel set of tempo-related audio features for application in audio retrieval is discussed, and it relies on the definition of the

[6] Tone is typically related to the brightness and the bandwidth of sound.

cyclic beat spectrum. A stochastic method for the representation of sung melodic contour is proposed in [65]: such a representation is obtained by fitting probability distribution functions. Semantic music retrieval by the use of social tags and Web-mined documents is also investigated in [10]: here feature kernel combination is used. In [24], an approach for automatic extraction of high-level audio features through distributed computing is presented, and an audio analysis method to create high-level annotation is discussed in [72].

Several works have also been proposed in the field of audio classification. The work of Liu [56], for example, represents a first attempt at building a multimedia search engine for the Internet: a fuzzy inference system for audio classification and retrieval. Support vector machines (SVMs) have also been employed with a multi-classification strategy [40] to address this problem. Experiments with SVM methods showed good results, an example of effective training for audio categorization with low error rate. In [60, 71], Gaussian Mixture Models (GMM) were used to classify audio by grouping N-dimensional feature vectors. These approaches involve the use of different kinds of standard features (MFCC, loudness, sharpness, etc.) as well as original features (entropy modulation, stationary segment duration, and so on). A hidden Markov model-based approach is presented in [76], where acoustic segment models are used to classify effectively music genres. Recently, some hybrid approaches have been proposed for audio classification. In [18], a weight factor based on support vector is applied to the Euclidean distance and k-NN rule to improve accuracy by 28 % with respect to the classic Euclidean-based k-NN classifier. The modified two-dimensional root cepstrum was introduced in [46] and tested to classify recordings of different classes of gunshots: an accuracy of 98.93 % was reported.

14.4.4 Applications and Tools for Content-Based Audio Retrieval

Different researches in the field of audio retrieval have focused on the development of applications and tools. Until now, the most successful application is Shazam [90]. The algorithm, introduced by Wang in 2003, relies on the fingerprinting technique and combinatorial hashing. It has proven to be robust against noise and distortions, and its computational efficiency allows great scalability; in fact, it is the most used audio search algorithm for mobile audio search engines. Experimental results on a database of about 20,000 tracks showed a search time of about 5–500 ms. Early tools for audio retrieval date back to 2000. Works such as Maryas, LibXtract, MIRtoolbox, and Mirage [13, 54, 82] relied on fast Fourier transform (FFT) algorithms to analyze audio content, extract features, and optimize audio similarity. MARSYAS[7] is a software framework for rapid prototyping and experimentation

[7] http://marsyas.info

with audio analysis and synthesis with specific emphasis to music signals and music information retrieval. The basic goal is to provide a general, extensible, and flexible architecture that allows easy experimentation with algorithms and provides fast performance that is useful in developing real-time audio analysis and synthesis tools. LIBXTRACT[8] is a simple, portable, lightweight library of audio feature extraction functions. The purpose of the library is to provide a relatively exhaustive set of feature extraction primitives that are designed to be "cascaded" to create extraction hierarchies. For example, "variance," "average deviation," "skewness," and "kurtosis" all require the "mean" of the input vector to be precomputed. MIRtoolbox[9] is a MATLAB toolbox dedicated to the extraction of musical features from audio files, including routines for statistical analysis, segmentation, and clustering. MIRtoolbox integrates a user-friendly syntax that enables to easily combine low- and high-level operators into complex flowcharts. MIRAGE[10] is an implementation of the research in automatic playlist generation and music similarity. Mirage analyzes music collection and computes acoustic similarity models for each song. It is also able to automatically generate playlists of similar music. It implements psychoacoustic modeling and Gaussian models as well as the FFT algorithm. RythMiXearch is a project presented by Kato that implements a query by example approach to music search. This method is able to accept two examples for each query and mix them; then, using latent Dirichlet allocation, it builds clusters and retrieves similar objects.

14.5 Further Application Domains for Content-Based Multimedia Retrieval

In the last section, we show possible applications of previously discussed methodologies, especially in the cultural heritage domain that represents in Italy a resource of inestimable value. The Internet of Things and Internet of Services are becoming the basic building blocks toward a unified ICT platform for a variety of applications. Since cellular phones and participatory sensor networks easily allow public and professional users to analyze and share local knowledge, there is an increasing interest in applying outcomes from information retrieval research to mobile applications. Several works have been proposed in this regard.

In [4], a novel data model for 3D objects and a tool for building, querying, and storing 3D objects into a relational database are presented. The preliminary environment is designed to be powerful enough to include functionalities and features of modern 3D description languages (X3D and Collada).

[8] http://libxtract.sourceforge.net/

[9] http://www.mathworks.com/matlabcentral/fileexchange/24583-mirtoolbox

[10] http://hop.at/mirage/

A multimedia semantic approach for recommending item in browser system, based on semantic contents and low-level features, is proposed in [1]. Here, a multimedia object recommender prototype for browsing the "Uffizi Gallery" digital picture collection is implemented to investigate the effectiveness of the proposed approach, based on users' satisfaction. The recommender system aims at helping users browse digital reproduction of Uffizi Gallery by suggestions computed with a novel method for recommendations.

Another interesting application is introduced in [5] by the DATABENC district. The aim of the project is to customize the system for an indoor museum, providing a personalized visiting experience for tourists (e.g., by designing specific "talking objects").

In the SNOPS project for smart cities, the focus is on the context-aware recommendation services. A recommendation strategy for planning browser activities exploiting object features, user behaviors, and context information is described in [3]. The contextual knowledge is modeled by using ontologies similar to CONON [91].

In this work, a system that provides personalized visiting paths to the tourists of Herculaneum ruins is presented. The effectiveness of this approach is evaluated by investigating the browsing effectiveness and users' satisfaction.

In this chapter, we have described how different information retrieval approaches can be used and mixed in order to build complex pervasive information systems that enhance user experience. These techniques find wide application in several emerging contexts such as smart cities.

References

1. Albanese, M., d'Acierno, A., Moscato, V., Persia, F., Picariello, A.: A multimedia semantic recommender system for cultural heritage applications. In: 2011 Fifth IEEE International Conference on Semantic Computing (ICSC), pp. 403–410. IEEE, Palo Alto (2011)
2. Amato, F., Mazzeo, A., Penta, A., Picariello, A.: Building rdf ontologies from semi-structured legal documents. In: CISIS, pp. 997–1002 (2008)
3. Amato, F., Chianese, A., Moscato, V., Picariello, A., Sperli, G.: Snops: a smart environment for cultural heritage applications. In: Proceedings of the Twelfth International Workshop on Web Information and Data Management, pp. 49–56. ACM, Maui (2012)
4. Amato, F., Mazzeo, A., Moscato, V., Picariello, A.: Building and retrieval of 3d objects in cultural heritage domain. In: 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 816–821. IEEE, Palermo (2012)
5. Amato, F., Chianese, A., Mazzeo, A., Moscato, V., Picariello, A., Piccialli, F.: The talking museum project. Proc. Comput. Sci. **21**, 114–121 (2013)
6. Ardizzone, E., La Cascia, M.: Automatic video database indexing and retrieval. In: Zhang, H., Aigrain, P., Petkovic, D. (eds.) Representation and Retrieval of Video Data in Multimedia Systems, pp. 29–56. Springer, New York (1997). doi:10.1007/978-0-585-31786-13. http://dx.doi.org/10.1007/978-0-585-31786-13
7. Arman, F., Hsu, A., Chiu, M.Y.: Image processing on encoded video sequences. Multimedia Syst. **1**(5), 211–219 (1994). doi:10.1007/BF01268945. http://dx.doi.org/10.1007/BF01268945

8. Bach, J.R., Fuller, C., Gupta, A., Hampapur, A., Horowitz, B., Humphrey, R., Jain, R.C., Shu, C.F.: Virage image search engine: an open framework for image management. In: Storage and Retrieval for Image and Video Databases (1996). doi:10.1117/12.234785. http://dx.doi.org/10.1117/12.234785
9. Bahrami, A., Jun, Y.: Methods and systems for context based query formulation and information retrieval. U.S. Patent No. 7,970,786. U.S. Patent and Trademark Office, Washington, DC, June 2011
10. Barrington, L., Yazdani, M., Turnbull, D., Lanckriet, G.R.: Combining feature kernels for semantic music retrieval. In: ISMIR, pp. 614–619 (2008)
11. Bolasco, S.: Statistica testuale e text mining: alcuni paradigmi applicativi. Quaderni di Statistica **7**, 17–53 (2005)
12. Bordogna, G., Pasi, G.: A fuzzy linguistic approach generalizing boolean information retrieval: a model and its evaluation. J. Am. Soc. Inf. Sci. **44**(2), 70–82 (1993)
13. Bullock, J.: libxtract: a lightweight library for audio feature extraction. In: Proceedings of the 2007 International Computer Music Conference, vol. 2, pp. 25–28. ICMA, Copenhagen (2007)
14. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Comput. Surv. (CSUR) **44**(1), 1 (2012)
15. Chang, S.K., Liu, S.H.: Picture indexing and abstraction techniques for pictorial databases. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-6**(4), 475–484 (1984). doi:10.1109/TPAMI.1984.4767552
16. Chang, S., Chen, W., Meng, H.J., Sundaram, H., Zhong, D.: Videoq: an automated content based video search system using visual cues. In: Proceedings of ACM Multimedia, pp. 313–324 (1997)
17. Chen, J.Y., Taskiran, C., Delp, E., Bouman, C.: Vibe: a new paradigm for video database browsing and search. In: Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries, 1998, pp. 96–100 (1998). doi:10.1109/IVL.1998.694510
18. Chung, Y.Y., Choi, E.H.C., Liu, L., Shukran, M.A.M., Shi, D.Y., Chen, F.: A new hybrid audio classification algorithm based on svm weight factor and euclidean distance. In: Proceedings of the 2007 Annual Conference on International Conference on Computer Engineering and Applications, CEA'07, pp. 152–157. World Scientific and Engineering Academy and Society, Stevens Point (2007). http://dl.acm.org/citation.cfm?id=1348258.1348286
19. Clarizia, F., Colace, F., De Santo, M., Greco, L., Napoletano, P.: Mixed graph of terms for query expansion. In: 11th International Conference on Intelligent Systems Design and Applications (ISDA), 2011, pp. 581–586 (2011). doi:10.1109/ISDA.2011.6121718
20. Colace, F., Santo, M.D., Greco, L.: An adaptive product configurator based on slow intelligence approach. Int. J. Metadata Semant. Ontol. **9**(2), 128–137 (2014). doi:10.1504/IJMSO.2014.060340. http://dx.doi.org/10.1504/IJMSO.2014.060340
21. Colace, F., Santo, M.D., Greco, L., Napoletano, P.: Text classification using a few labeled examples. Comput. Hum. Behav. (2014). doi:10.1016/j.chb.2013.07.043. http://www.sciencedirect.com/science/article/pii/S0747563213002823
22. Cummins, R.: A standard document score for information retrieval. In: Proceedings of the 2013 Conference on the Theory of Information Retrieval, p. 24. ACM, New York (2013)
23. D'Agostino, E., Elia, A., Vietri, S.: Lexicon-grammar, electronic dictionaries and local grammars of Italian. In: Leclère, Ch., Laporte, É., Piot, M., Silberztein, M. (eds.) Lexique, Syntaxe et Lexique-Grammaire. Papers in Honour of Maurice Gross, Lingvisticae Investigationes Supplementa, vol. 24. IGML, Amsterdam (2004)
24. Deliege, F., Chua, B.Y., Pedersen, T.B.: High-level audio features: distributed extraction and similarity search. In: Ninth International Conference on Music Information Retrieval, Philadelphia, pp. 565–570, September 2008
25. Downie, J.S.: Music information retrieval. Annu. Rev. Inf. Sci. Technol. **37**(1), 295–340 (2003). doi:10.1002/aris.1440370108. http://dx.doi.org/10.1002/aris.1440370108
26. Eakins, J., Graham, M.: Content-based image retrieval, Technical Report (1999)
27. Edmunds, A., Morris, A.: The problem of information overload in business organisations: a review of the literature. Int. J. Inf. Manag. **20**(1), 17–28 (2000)

28. Egozi, O., Markovitch, S., Gabrilovich, E.: Concept-based information retrieval using explicit semantic analysis. ACM Trans. Inf. Syst. (TOIS) **29**(2), 8 (2011)
29. Elia, A., Vietri, S., Postiglione, A., Monteleone, M., Marano, F.: Data mining modular software system. In: SWWS, pp. 127–133 (2010)
30. Elia, A., Guglielmo, D., Maisto, A., Pelosi, S.: A linguistic-based method for automatically extracting spatial relations from large non-structured data. In: Algorithms and Architectures for Parallel Processing, pp. 193–200. Springer, Heidelberg (2013)
31. Fablet, R., Bouthemy, P., Pérez, P.: Statistical motion-based video indexing and retrieval. In: International Conference on Content-Based Multimedia Information Access, pp. 602–619 (2000)
32. Faloutsos, C., Oard, D.W.: A survey of information retrieval and filtering methods. Technical Report, CS-TR-3514, University of Maryland (1995)
33. Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., Equitz, W.: Efficient and effective querying by image content. J. Intell. Inf. Syst. **3**(3-4), 231–262 (1994). doi:10.1007/BF00962238. http://dx.doi.org/10.1007/BF00962238
34. Fan, J., Elmagarmid, A.K., Zhu, X., Aref, W.G., Wu, L.: Classview: hierarchical video shot classification, indexing, and accessing. Trans. Multimedia **6**(1), 70–86 (2004). doi:10.1109/TMM.2003.819583. http://dx.doi.org/10.1109/TMM.2003.819583
35. Fernández, M., Cantador, I., López, V., Vallet, D., Castells, P., Motta, E.: Semantically enhanced information retrieval: an ontology-based approach. Web Semant. Sci. Serv. Agents World Wide Web **9**(4), 434–452 (2011)
36. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P.: Query by image and video content: the qbic system. Computer **28**(9), 23–32 (1995). doi:10.1109/2.410146
37. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJCAI, vol. 7, pp. 1606–1611 (2007)
38. Gantz, J., Reinsel, D.: Extracting value from chaos. IDC iview, pp. 1–12 (2011)
39. Grossman, D.A.: Information Retrieval: Algorithms and Heuristics, vol. 15. Springer, New York (2004)
40. Guo, G., Li, S.Z.: Content-based audio classification and retrieval by support vector machines. IEEE Trans. Neural Netw. **14**(1), 209–215 (2003)
41. Gupta, A., Jain, R.: Visual information retrieval. Commun. ACM **40**(5), 70–79 (1997). doi:10.1145/253769.253798. http://doi.acm.org/10.1145/253769.253798
42. Han, S., Yoon, K., Kweon, I.: A new technique for shot detection and key frames selection in histogram space. In: 12th Workshop on Image Processing and Image Understanding, pp. 475–479 (2000)
43. Harris, Z.S.: Linguistic transformations for information retrieval. In: Papers in Structural and Transformational Linguistics, pp. 458–471. Springer, Dordrecht (1970)
44. Hliaoutakis, A., Varelas, G., Voutsakis, E., Petrakis, E.G., Milios, E.: Information retrieval by semantic similarity. Int. J. Semant. Web Inf. Syst. **2**(3), 55–73 (2006)
45. Hopcroft, J.E.: Introduction to Automata Theory, Languages, and Computation. Pearson Education India, New Delhi (1979)
46. Hossein, M.: Automatic audio classification using modified two dimensional root cepstral features. In: The International Conference on Electrical Engineering (2008)
47. Ishikawa, Y., Subramanya, R., Faloutsos, C.: Mindreader: querying databases through multiple examples. Computer Science Department, p. 551 (1998)
48. Jain, R.: Dynamic vision. In: 9th International Conference on Pattern Recognition, 1988, vol. 1, pp. 226–235 (1988). doi:10.1109/ICPR.1988.28212
49. Kara, S., Alan, Ö., Sabuncu, O., Akpınar, S., Cicekli, N.K., Alpaslan, F.N.: An ontology-based retrieval system using semantic indexing. Inf. Syst. **37**(4), 294–305 (2012)
50. Klapuri, A., Davy, M. (eds.): Signal Processing Methods for Music Transcription. Springer, New York (2006)

51. Kurth, F., Gehrmann, T., Müller, M.: The cyclic beat spectrum: tempo-related audio features for time-scale invariant audio identification. In: Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR), Victoria, pp. 35–40, 8–12 October 2006
52. Landauer, T.K., Dumais, S.T.: A solution to plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychol. Rev. **104**(2), 211 (1997)
53. Landauer, R., Swanson, J.: Frequency factors in the thermally activated process. Phys. Rev. **121**(6), 1668 (1961)
54. Lartillot, O., Toiviainen, P.: A matlab toolbox for musical feature extraction from audio. In: International Conference on Digital Audio Effects, pp. 237–244 (2007)
55. Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based multimedia information retrieval: state of the art and challenges. ACM Trans. Multimedia Comput. Commun. Appl. **2**(1), 1–19 (2006). doi:10.1145/1126004.1126005. http://doi.acm.org/10.1145/1126004.1126005
56. Liu, M., Wan, C., Wang, L.: Content-based audio classification and retrieval using a fuzzy logic system: towards multimedia search engines. Soft Comput. **6**(5), 357–364 (2002). doi:10.1007/s00500-002-0189-3. http://dx.doi.org/10.1007/s00500-002-0189-3
57. Ma, W., Manjunath, B.: Netra: a toolbox for navigating large image databases. In: Proceedings of International Conference on Image Processing, 1997, vol. 1, pp. 568–571 (1997). doi:10.1109/ICIP.1997.647976
58. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
59. Massoudi, K., Tsagkias, M., de Rijke, M., Weerkamp, W.: Incorporating query expansion and quality indicators in searching microblog posts. In: Advances in Information Retrieval, pp. 362–367. Springer, Heidelberg (2011)
60. Mckinney, M., Breebaart, J.: Features for audio and music classification. In: Proceedings of the International Symposium on Music Information Retrieval, pp. 151–158 (2003)
61. Mehrotra, S., Rui, Y., Ortega-Binderberger, M., Huang, T.: Supporting content-based queries over images in mars. In: Proceedings of IEEE International Conference on Multimedia Computing and Systems '97, pp. 632–633 (1997). doi:10.1109/MMCS.1997.609791
62. Mitrovic, D., Zeppelzauer, M., Breiteneder, C.: Features for content-based audio retrieval. Adv. Comput. **78**, 71–150 (2010). http://dblp.uni-trier.de/db/journals/ac/ac78.htmlMitrovicZB10
63. Mojsilovic, A., Rogowitz, B.: Capturing image semantics with low-level descriptors. In: Proceedings of 2001 International Conference on Image Processing, vol. 1, pp. 18–21 (2001). doi:10.1109/ICIP.2001.958942
64. Ogle, V.E., Stonebraker, M.: Chabot: retrieval from a relational database of images. Computer **28**(9), 40–48 (1995)
65. Ohishi, Y., Goto, M., Itou, K., Takeda, K.: A stochastic representation of the dynamics of sung melody. In: Dixon, S., Bainbridge, D., Typke, R. (eds.) ISMIR, pp. 371–372. Austrian Computer Society, Vienna (2007). http://dblp.uni-trier.de/db/conf/ismir/ismir2007.htmlOhishiGIT07
66. Papathomas, T., Conway, T., Cox, I., Ghosn, J., Miller, M., Minka, T., Yianilos, P.: Psychophysical studies of the performance of an image database retrieval system. In: IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, Conference on Human Vision and Electronic Imaging III, pp. 591–602 (1998)
67. Patel, A.J.: Systems and methods for highlighting search results. U.S. Patent No. 6,839,702, 4 Jan 2005
68. Pentland, A., Picard, R., Sclaroff, S.: Photobook: content-based manipulation of image databases. Int. J. Comput. Vis. **18**(3), 233–254 (1996). doi:10.1007/BF00123143. http://dx.doi.org/10.1007/BF00123143
69. Picard, R., Minka, T.: Vision texture for annotation. Multimedia Syst. **3**(1), 3–14 (1995). doi:10.1007/BF01236575. http://dx.doi.org/10.1007/BF01236575
70. Pickard, A.J.: Research Methods in Information. Facet, London (2013)
71. Pinquier, J., Arias, J., André-Obrecht, R.: Audio classification by search of primary components. In: International Workshop on Image, Video and Audio Retrieval and Mining, Sherbrooke (2004)

72. Pohle, T., Knees, P., Seyerlehner, K., Widmer, G.: A high-level audio feature for music retrieval and sorting. In: DAFx-10 (2010)
73. Quinlan, J.R.: Combining instance-based and model-based learning. In: ICML, pp. 236–243 (1993)
74. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice-Hall, Upper Saddle River (1993)
75. Ranchhod, E.: Lexique-grammaire du portugais: prédicats nominaux supportés par estar. Lingvisticae Investigationes **13**(2), 351–367 (1989)
76. Reed, J.: A study on music genre classification based on universal acoustic models. In: ISMIR 2006, pp. 89–94 (2006)
77. Roa-Valverde, A.J., Sicilia, M.A.: A survey of approaches for ranking on the web of data. Inf. Retr. **17**, 295–325 (2014)
78. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. J. Am. Soc. Inf. Sci. **27**(3), 129–146 (1976)
79. Rogowitz, B.E., Frese, T., Smith, J., Bouman, C.A., Kalin, E.: Perceptual image similarity experiments. In: Proceedings of the SPIE Human Vision and Electronic Imaging III, vol. 3299 (1998)
80. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
81. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
82. Schnitzer, D.: High-performance music similarity computation and automatic playlist generation. Technical Report, University of Technology, Vienna (2007)
83. Scholtes, J.C.: Unsupervised learning and the information retrieval problem. In: 1991 IEEE International Joint Conference on Neural Networks, pp. 95–100. IEEE, Seattle (1991)
84. Sethi, I.K., Coman, I.L., Stan, D.: Mining association rules between low-level image features and high-level concepts (2001). doi:10.1117/12.421083. http://dx.doi.org/10.1117/12.421083
85. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell. **22**(12), 1349–1380 (2000)
86. Smith, J.R., Chang, S.F.: Visualseek: a fully automated content-based image query system. In: Proceedings of the Fourth ACM International Conference on Multimedia, MULTIMEDIA ’96, pp. 87–98. ACM, New York (1996). doi: 10.1145/244130.244151. http://doi.acm.org/10.1145/244130.244151
87. Tamine-Lechani, L., Boughanem, M., Daoud, M.: Evaluation of contextual information retrieval effectiveness: overview of issues and research. Knowl. Inf. Syst. **24**(1), 1–34 (2010)
88. Tzanetakis, G., Ermolinskyi, A., Cook, P.: Pitch histograms in audio and symbolic music information retrieval. In: Proceedings of the Third International Conference on Music Information Retrieval: ISMIR, pp. 31–38 (2002)
89. Wang, J., Li, J., Wiederhold, G.: Simplicity: semantics-sensitive integrated matching for picture libraries. IEEE Trans. Pattern Anal. Mach. Intell. **23**(9), 947–963 (2001). doi:10.1109/34.955109
90. Wang, A., et al.: An industrial strength audio search algorithm. In: ISMIR, pp. 7–13 (2003)
91. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–22. IEEE, Orlando (2004)
92. Wendlandt, E.B., Driscoll, J.R.: Incorporating a semantic analysis into a document retrieval strategy. In: Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 270–279. ACM, New York (1991)
93. Wold, E., Blum, T., Keislar, D., Wheaten, J.: Content-based classification, search, and retrieval of audio. IEEE MultiMedia **3**(3), 27–36 (1996). doi:10.1109/93.556537

94. Zabih, R., Miller, J., Mai, K.: A feature-based algorithm for detecting and classifying scene breaks. In: Proceedings of the Third ACM International Conference on Multimedia, MULTIMEDIA '95, pp. 189–200. ACM, New York (1995). doi:10.1145/217279.215266. http://doi.acm.org/10.1145/217279.215266

95. Zhou, X.S., Huang, T.S.: Cbir: from low-level features to high-level semantics. In: Proceedings of SPIE Image and Video Communication and Processing (2000). doi:10.1117/12.382975. http://dx.doi.org/10.1117/12.382975

Chapter 15
Multimedia Queries in Digital Libraries

Ilaria Bartolini and Marco Patella

15.1 Peculiarities of Querying Multimedia Data

As has been described in [1], multimedia queries are entirely different with respect to those commonly used in traditional information retrieval and/or database systems. Indeed, the diverse content and complex structure of multimedia documents make traditional (Boolean) queries much less effective. As an example, in a digital image library, we may want to retrieve pictures that are similar in content to a user-provided image: The result we expect from the system is probably a set of library images, sorted for decreasing values of visual similarity with respect to the query; this is very different to the result of a regular SELECT query.

The processing of queries in multimedia digital libraries should also take into account the distinctive features of multimedia data, so that one considers the following aspects:

- Efficiency, due to the size and amount of multimedia data which exceed those of textual data by several orders of magnitude
- Effectiveness, since the representation of the "local" information to the system may differ from what the user issuing the query has in mind
- Semantics, due to the fact that "concepts" can be attached (implicitly or as metadata) to each multimedia datum
- Dynamicity, because digital content is continuously added to/removed from the library
- Complexity, since a multimedia document is usually composed of several parts, each of a particular medium (e.g., a Web page contains text, images, videos, and so on)

I. Bartolini (✉) • M. Patella
Department of Computer Science and Engineering – DISI, University of Bologna, Bologna, Italy
e-mail: i.bartolini@unibo.it; marco.patella@unibo.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_15

The usual approach pursued by digital libraries is to represent information about documents as metadata that are harvested when each single document enters the system. Then, such metadata are "indexed" to allow the retrieval of query results: This can be carried out by exploiting inverted files. In this way, the digital library is not accessed during the search, making the whole process very efficient. The above approach, however, fails to address the problems of effectiveness, since the metadata harvesting process is hardly tailored to the needs of a specific user, and complexity, mainly because the library datum is commonly seen as an atomic entity, which cannot be divided into subparts.

The typical query paradigms that are offered by multimedia digital libraries are of two types:

Boolean search: The user specifies a (complex) Boolean predicate on metadata, and only those documents whose metadata satisfy the predicate are returned. The result of such a query is an (unordered) set of objects.

Ranked search: The user provides a multimedia object, and library data are returned in descending order of similarity with respect to the query. The result of this query is an (ordered) list of objects. The user has three ways of limiting the result size:

> Range query: The user specifies a minimum similarity threshold θ, so that objects whose similarity with the query is lower than θ are not returned (note that this requires a knowledge of the distribution of similarity values between objects).
>
> k-NN query: The user provides a maximum number of results *k*, and only the *k* objects more similar to the query are returned.
>
> Sorted access: Library data are returned to the user in descending order of similarity with respect to the query, until the user is satisfied with the result (similar to a common iterator).

If one wants to support more complex query types, the main challenge is how the system can assess whether or not a datum is relevant for the query.

Consider, for example, a digital library offering access to documentaries. Every movie also contains metadata like year of production, title, name of the director, etc. The system provides the users visiting the library with two different search paths:

1. A search-by-content paradigm, where the user can provide a sample video and library movies are returned in descending order of visual similarity with respect to the provided video.
2. A search-by-metadata paradigm, expressed by means of a Boolean predicate on metadata, e.g., director = "Riefenstahl" AND year BETWEEN 1920 AND 1929.

(continued)

What if the user wants only the documentaries shot by Leni Riefenstahl during the 1920s but sorted for visual similarity with respect to a provided video? How can such a query be processed by the system? Should a documentary with very similar content but very different metadata be included in the result? In case how such documentary should be ranked with respect to another movie with different visual content but satisfying the Boolean predicate?

In the following text, we will discuss how modern systems for the management of digital libraries can be extended to tackle the above-described problems. First, we present a model able to capture the complexity and diversity of multimedia documents and show how such a model can also support efficient query resolution (Sect. 15.2). Then, we face the problem of enriching multimedia documents with semantic annotations (Sect. 15.3). It is widely known, in fact, that content-based multimedia retrieval using low-level features only does not provide sufficiently accurate results. This is due to the so-called semantic gap existing between the user-subjective notion of similarity and the one implemented by the system. The use of semantic annotations, therefore, allows querying digital libraries using high-level semantic concepts, helping to bridge the semantic gap.

15.2 The Windsurf Model

In the Windsurf model [8],[1] each multimedia document D is composed of n_D elements, $D = \{R_1, \ldots, R_{n_D}\}$. Each element can be recursively defined in terms of other components and so on (e.g., a video can be represented by a set of shots, a shot by a set of frames, a frame by a set of image regions, etc.). Simple (base) elements are finally described by way of metadata, including features that allow the comparison of elements. Note that the subdivision of a multimedia document into its component parts can be either explicit (i.e., specified in the very same description of the multimedia document, like in a Web page) or implicit. In the latter case, it is the system that automatically breaks the multimedia object into component parts; examples of this automatic subdivision include the following:

- A video segmented into scenes of homogeneous visual content [9]
- An image divided into regions of pixels sharing common color and/or texture [2, 8]
- An image represented by way of its salient points [10, 19]

When compared to query objects Q, also composed of query elements $Q = \{Q_1, \ldots, Q_n\}$, the components of D are compared to those of Q by way of a specific similarity function, and the overall similarity $s(Q, D)$ (recursively) depends on the

[1] www-db.disi.unibo.it/Windsurf/

similarity between components. This modular approach, which we borrowed from the PANDA model [6] for describing mining patterns, allows for a great generality and flexibility. Indeed, although for comparing complex multimedia documents one could devise arbitrary models, it is useful and, at the same time, efficient for practical purposes to consider solutions that decompose the "difficult" problem of comparing complex documents into simpler subproblems like those of comparing simple components and then "smartly" aggregating the so-obtained partial solutions into an overall similarity value. Comparing complex documents is achieved by combining two basic elements:

1. The matching type, which is used to establish how the component elements of the two multimedia documents can be matched
2. The aggregation logic, which is used to combine the similarity scores of the matched component elements into a single value representing the total similarity between the complex documents

The matching type specifies the constraints that should be used when comparing components: For example, it is obvious that only components of the same type are to be compared (i.e., one cannot compare a video with a text); otherwise, one can specify that only one-to-one matchings are allowed (e.g., the same image in a multimedia document cannot be matched to two different images in another document). The aggregation logic, on the other hand, states how scores of the matched component pairs should be aggregated so as to yield the overall similarity score. We note here that among all the valid matchings (as specified by the matching type), the rationale is to pick the "best" one, i.e., the one that maximizes the aggregated similarity [6].

The choice of which matching type and aggregation logic are better suited for the query at hand can be left to the user or assumed by default, with the understanding that this choice has a clear impact on both the effectiveness and the efficiency of the result.

We finally note that an appropriate choice of matching type and/or aggregation logic can also support other query types, like those denoted in [7] as *partial*, *zoom-in*, *containment*, and *part-of* queries, i.e., queries where only a fraction of components of the query and/or of the library document are considered.

The flexibility and generality of the proposed model are indeed required to deal with the possible complexity of queries in multimedia digital libraries.

As another example, we consider a digital library storing Web pages about music composers. Every page includes a textual part containing structured data, like the composer name, his nationality, his date of birth, and so on, and an unstructured part containing other documents, like pictures, other texts, and audio fragments, all of them possibly tagged with metadata.

A user might be interested in all documents related to German composers of the nineteenth century that contain a picture sufficiently similar to a user-specified image and an audio fragment at least 30 s long. Such a query contains both predicates on metadata (nationality, date of birth, length of the audio fragment) and similarity predicates (similar image). Moreover, some of the predicates are related to the whole document (nationality, date of birth), while others only concern component parts (length of the audio fragment, similar image). It is clear that with a reductionist approach that considers a document as composed of parts, the complexity of such a query is easily dealt with, as opposed to a holistic strategy that considers the multimedia datum as an atomic entity.

15.2.1 Efficient Processing of Similarity Queries

Having defined the ingredients needed to assess the similarity between two multimedia objects, we then derive a sequential algorithm able to retrieve the best results for a given query. Clearly, any sequential algorithm will incur prohibitively high costs even for moderately large digital libraries. In order to guarantee scalability, the query processor should exploit the presence of indices able to efficiently retrieve relevant results. Our arguments will be developed independently of the specific index; rather, we will refer to a generic distance-based index, i.e., any index that relies on the computation of distances to return back objects. Distance-based indices include both multidimensional [15] and metric [11] indices, relevant examples of which are the R-tree [16] and the M-tree [12], respectively.

The resolution of a similarity query is depicted in Fig. 15.1. After the query document has been decomposed into its component parts, each query part is processed separately by a distance-based index using a sorted access. The results of each index, a list of objects ranked for decreasing similarity with respect to the input query component, are then fed to the middleware algorithm implementing the chosen matching/aggregation strategy. With the help of a *random access* to the database of document components, the middleware algorithm is able to compute the similarity value for all those objects for which at least a component has been returned by at least an index scan. As soon as the algorithm is able to assess that yet unseen documents cannot lead to a similarity score which is higher than the "best" document seen so far, then such a document can be returned to the user

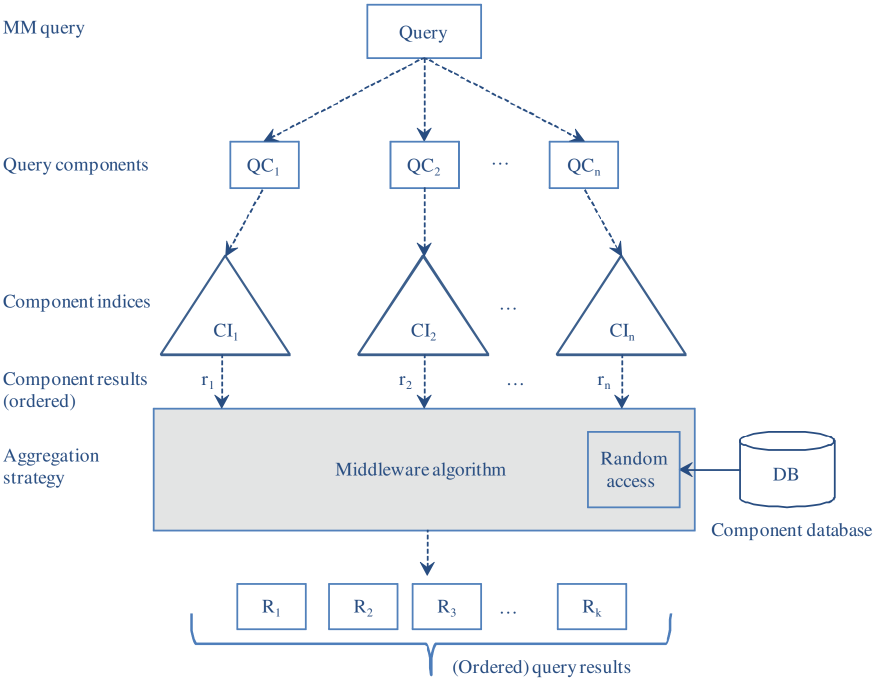


Fig. 15.1 Processing a similarity query: the middleware algorithm combines, through the matching/aggregation strategy, sorted accesses to the component indices and random accesses to the component database. The result is a sorted list of library documents, ranked for decreasing values of similarity to the query

with all subsequent documents in the list. The above can be guaranteed [7, 13] if the aggregation logic is monotone, i.e., a lower similarity score for any component cannot increase the overall similarity score, a property which is sound and is shared by all the most used aggregation logics.

When considered as a whole, Fig. 15.1 can be seen as a (complex) index which is able to support sorted access, since results are provided in a ranked order. Therefore, this can be recursively used to obtain the arbitrary complexity required by the model: Distance-based indices are used on simple (base) components, while the complex index depicted in Fig. 15.1 is used on complex multimedia objects.

15.2.2 Processing of Mixed Queries

Although the strategy described in the previous section allows us to efficiently process similarity queries, it does not help when mixed queries, i.e., queries combining metadata and similarity predicates, are issued. For such queries, we have to somehow combine the result of a similarity query (an ordered list of objects) with that of a metadata query (a set of objects). The possibilities here are those listed in [5]: join with order, union with order, and difference with order; we focus here on the most complex operator, join with order, since the implementation of other operators does not differ much from those commonly used for relational DBs.

To exemplify our arguments, we use again the example on documentaries. The metadata predicate (director = “Riefenstahl” AND year BETWEEN 1920 AND 1929) would return a set of documentaries satisfying both Boolean conditions, while for the similarity query, three possibilities exist:

1. If the similarity predicate is a range query, the user has also specified a minimum similarity threshold θ; the similarity predicate is now analogous to a Boolean predicate, $s(Q, D) \geq \theta$. The query result should therefore include only those documentaries that satisfy all three conditions, sorted for decreasing values of similarity.
2. If the similarity predicate is a k-NN query, the first possibility is to consider the maximum number of results k specified by the user as an a priori condition; the similarity predicate is again analogous to a Boolean predicate, which is satisfied only by the k movies most similar to the provided one. Again, the query result includes only those documentaries that satisfy all three conditions, sorted for decreasing values of similarity (note that this will retrieve a number of results which is not higher than k but can also produce an empty result if the k best movies do not satisfy the metadata predicate).
3. Finally, we can consider k as an a posteriori condition, i.e., the user wants the k documentaries shot by Leni Riefenstahl during the 1920s ranked for visual similarity with respect to the provided video (assuming that k of such objects exist).

Figure 15.2 depicts the components of the mixed query processor: The two query predicates (MQC, multimedia query component, and SQC, semantic query component) are fed to the relative indices (MI, multimedia/similarity index, and SI, semantic/metadata index); the so-obtained results have to be combined by the matcher to produce the result for the overall query.

It is obvious that processing cases 1 and 2 above is quite simple, since one can compute the intersection of results provided by the similarity index and by the semantic index. For case 3 we have two alternative strategies:

1. The first strategy retrieves all documentaries satisfying the metadata predicate (by using SI) and sorts them by way of the similarity predicate. Then, only the first k results are returned to the user.
2. As an alternative, we can perform a sorted access on MI and check, for all retrieved documentaries, the metadata predicate, returning objects satisfying it until k results have been output.

The above strategies provide the same result but have different efficiency. In particular, this heavily depends on the selectivity f of the metadata predicate and on the cost of evaluating the similarity predicate. Since we will assume that the latter is

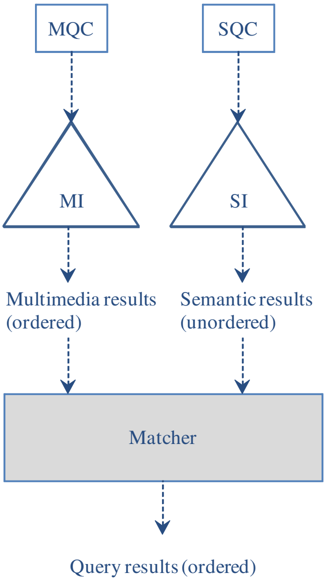


Fig. 15.2 Processing a mixed query: the results obtained by the similarity index (MI) and the semantic index (SI) are combined by the matcher to produce the result

always (much) higher than evaluating the metadata predicate, due to the complexity of comparing multimedia features [1], for high selective metadata predicates, the first strategy is to be preferred: In this case only a few documents ($f \times N$, where N is the number of documents in the library) are compared against the query. On the other hand, for low selective predicates, the second strategy attains the best efficiency, because only k/f sorted accesses are required to solve the query (every retrieved document has a probability f to satisfy the metadata predicate). We conclude this section by highlighting the analogy with the optimizer component of a DBMS that has to choose the optimal query plan according to the (estimated) selectivity of query predicates.

15.3 Semantic Enrichment of Multimedia Data

Although content-based techniques presented above, possibly assisted with user relevance feedback from the user [4, 23], can indeed attain very good effectiveness, in several cases they still stay below the optimal 100 % precision value, in particular when the user is looking for documents matching some high-level concept, which is hardly representable by means of low-level features only. In such cases, a possible way to fill the semantic gap is to assign meaningful terms to documents, so as to indeed allow a high-level, concept-based retrieval.

Terms associated to documents can, indeed, be considered as a different type of metadata, one which is not included in the multimedia datum but which is

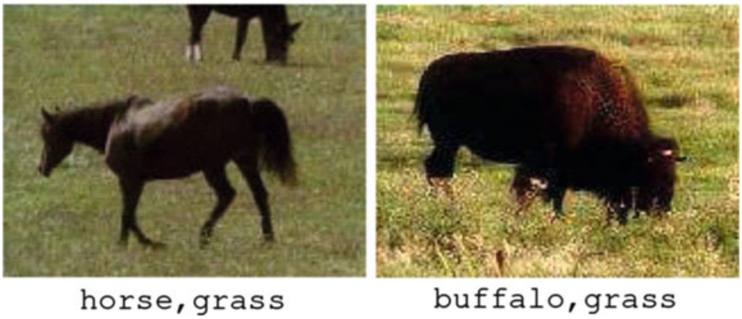


Fig. 15.3 Two images with associated tags

associated with it by way of a manual or automatic process. Clearly, such terms can be seamlessly used into predicates for Boolean search, as are traditional metadata.

For instance, assuming that the two images in Fig. 15.3 are labeled as shown, it would be possible to discriminate among them if, say, one is looking for horses and, at the same time, to consider both relevant if one is looking for mammals on grass.

Associating semantic labels to multimedia objects is usually performed in one of the following ways:

Tagging by an expert: This is the solution commonly used in libraries, where an expert provides labels for every datum. Such labels are expected to be of great quality, but the process is lengthy and expensive [1]; moreover, the problem of subjectivity can plague the whole data collection, because the user searching the library can have a different view on the data with respect to the user providing labels.

Social tagging: Using this approach, which is the one exploited by “social” libraries, like YouTube[2] and Flickr,[3] users accessing a multimedia document can also provide labels for it. The so-obtained annotation is usually of low quality, due to problems of ambiguity (because a label could carry different meanings due to polysemy or homonymy), lack of information (because a document could never have been labeled), and problems of synonymy/mistyping.

Automatic tagging: These techniques exploit the similarity among multimedia documents (computed by way of low-level features) to extract labels relevant for a non-annotated object, with the assumption that objects sharing similar features also convey the same semantic content and can thus be tagged with the same labels.

Several techniques for semiautomatic annotation of multimedia objects [3, 9] have been proposed in recent years, and the first prototypes for annotation are now

[2] www.youtube.com

[3] www.flickr.com

available on the Internet (e.g., ALIPR[4] and Behold[5] for images). We can group state-of-the-art solutions into two main classes, namely, semantic propagation and statistical inference. In both cases, the problem to be solved is the same: Given a training set of annotated multimedia objects, discover affinities between low-level features and terms that describe the object content, with the aim of predicting "good" terms to annotate a new document. With propagation models [20], a supervised learning technique that compares content similarity at a low level and then annotates objects by propagating terms over the most similar objects is adopted. Working with statistical inference models [22], an unsupervised learning approach tries to capture correspondences between low-level features and terms by estimating their joint probability distribution. Both approaches improve the annotation process and the retrieval on large multimedia libraries.

The typical approach for annotating multimedia objects exploits user-defined textual labels [3, 18, 24]. However, this is commonly performed by drawing tags from an unstructured set, thus not taking into account their intended meaning (i.e., the meaning such tags convey in the context where their associated multimedia data are found, which gives a sort of meaning vagueness to labels [21]). To deal with this, in order to connect each tag with its intended meaning, the coexistence of multiple, independent classification criteria [14] can be exploited. According to this "multidimensional" approach, labels belonging to different dimensions may have separate meanings, while each dimension will represent the meaning of high-level concepts contained therein, providing a disambiguation of their semantics.[6] Moreover, each dimension takes the shape of a tree, where each concept is represented within a taxonomy node and terms are linked with a parent/child relationship. More precisely, each concept is denoted as a *semantic tag*, represented as a path in a tree. To each tree node is therefore associated a single label; the label of the root node corresponds to the name of the dimension itself.

The above model of hierarchical faceted categories has been successfully used in a variety of applications to provide a coherent and complete description of data [9, 17]. For instance, Fig. 15.4 shows some dimensions for a real-world scenario: These include "animal," "landscape," "geographic location," and so on. Each node in a tree path corresponds to a more specialized concept with respect to its parent node, so that moving up/down within a tree, a user encounters more/less abstract semantic concepts. This means that if a document is tagged with a given semantic concept t, it is also associated to all ancestors of t. In the example of Fig. 15.4, the label "landscape/sky/rainbow" also includes the semantic concept "landscape/sky."

We finally note that, in line of principle, the same label could appear in several different trees: This allows to distinguish between the different uses and/or meanings

[4] www.alipr.com

[5] www.behold.cc

[6] Such dimensions are therefore quite different with respect to those used in other semantic indexing approaches, like bag-of-words or latent semantic indexing (LSI), where each dimension corresponds to a single, basic concept.

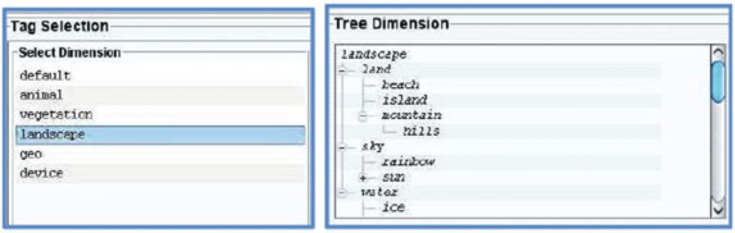


Fig. 15.4 A six-dimensional scenario: the "landscape" dimension is selected on the *left-hand side* and (part of) its concept tree is shown on the *right-hand side*

that different occurrences of the same label could convey. For example, it is possible that the label "turkey" will appear as a node within both the dimension "geographic location" (e.g., used to describe documents according to the location they were created) and the dimension "animal," associated to documents about the gallinaceous bird. Extending the example, we could conceive the existence of a "sports" dimension (associated to documents related to sports events), where the same "turkey" label could appear in several places, for example, "sports/soccer/turkey" and "sports/basketball/turkey," representing, respectively, the Turkish national soccer and basketball teams. Clearly, the fact that each semantic tag corresponds to a single complete path within a tree allows the disambiguation of the different uses of the same label, as illustrated by the previous example.

In order to ensure compatibility with unstructured ("flat") dimensions, such as those used in systems like YouTube or Flickr, we can support two-level taxonomies, with all tags appearing as children of the single tree root node (in Fig. 15.4, this is represented by the "default" dimension). This fact allows us to also exploit another technique to solve label ambiguity, i.e., label co-occurrence. On the other hand, label co-occurrence is not able by itself to solve problems of homonymy/polysemy. For example, suppose the user wants to retrieve documents about the animals of the Eurasian country of Turkey: It is quite likely that querying the system using the flat concepts "animal" and "turkey" would primarily return documents concerning the gallinaceous bird, due to the polysemy of the term "turkey." The system is therefore not able to satisfy this particular information need of the user by using label co-occurrence only.

According to the presented model, each document can be assigned a variable number of semantic tags. If a dimension is not relevant for a document, then no semantic tag from such a dimension is used to characterize the content. On the other hand, a document could be characterized by multiple semantic tags from the same dimension, if this is appropriate. For instance, an image depicting a dog and a cat might be assigned the two semantic tags "animal/dog" and "animal/cat," both from the "animal" dimension. Thus, although each dimension provides a means to classify documents, this classification is not exclusive at the instance level, a fact that provides the necessary flexibility to organize documents.

15.3.1 Efficient Annotation of Complex Multimedia Documents

For tagging purposes, we advocate the technique exploited in the imagination system [3], using a set of pre-annotated documents as a knowledge base. Such documents are used by the system as an example of the semantic concepts attached to them; in this way, only the concepts included in the knowledge base could possibly be suggested as relevant for a given (non-labeled) document. When provided with a document to be labeled, the system retrieves (by way of the multimedia index, MI) documents having a similar content and proposes semantic concepts depending on the similarity of the document with the documents in the knowledge base. The use of MI is clearly the key to attain efficiency in the annotation process. Every time a new document is processed and tagged, its information is also inserted into the semantic index (SI), hence improving the system accuracy and quality.

Although the above algorithm has proven effective for simple multimedia objects, e.g., for images, its accuracy on complex documents is questionable. Indeed, the model presented in Sect. 15.2 allows for an arbitrary complexity for a document D that can have components which have content similar to components of other documents but is still not very similar to any of the other documents in the library. Exploiting the above technique on the document D is likely to lead to an imprecise labeling of D. To overcome this problem for complex documents, we propose the use of hierarchical tagging [9].

The idea at the base of *hierarchical tagging* is quite simple: Labels associated to components of D are propagated to D using a frequency analysis (only most frequent labels are propagated to D). Propagating tags from components to documents is therefore an activity of summarization, i.e., the description of the document is a compact sum of the tags associated with its components.

Given a document D and a semantic tag t, the relevance of t for D is computed by combining the relevance of all components s in D (e.g., this could be the relative length of a scene in a video or of an audio track in a compilation or the size of an object in an image) and the relevance of t in s, which can vary from 0 to 1. More precisely, we denote the relevance of component s for D as $W(s,D)$ and the relevance of tag t in s as $A(t,s)$. Note that, in the default case, it is $W(s,D) = 1/n_D$, where n_D is the number of components of D, and $A(t,s) = 1$ for all tags t in s; otherwise, $A(t,s) = 0$. Then, the relevance of t in D can be defined as $R(t,D) = \sum_s A(t,s) \times W(s,D)$. Semantic tags can then be ranked for decreasing values of $R(t,D)$, and only the more relevant tags are selected for D. The same process can then be recursively applied upward in the hierarchy of multimedia documents.

As an example, we consider again the case of documentaries. Each video is (automatically or manually) divided into scenes, and a number of keyframes are automatically extracted from each scene, representing its visual content.

(continued)

Assume that each scene contained in the library has been annotated (either manually or automatically) and that a new documentary D is inserted in the library. D is then segmented into scenes, and the system extracts, for each scene, the relevant keyframes. Each keyframe f is then used to query the multimedia index and, say, the k most similar keyframes in the library are retrieved. The set of tags for f is then built using the tags of the scenes represented by the keyframes returned by the index.

Labels automatically suggested for frames are then propagated at the scene level: Here, $W(s, D)$ and $A(t, s)$ assume the default values of $1/n_D$ (with n_D denoting the number of keyframes representing a given scene) and 0/1. Finally, scene labels are summarized into tags for the whole video D, with $W(s, D)$ equal to the relative length of scene s with respect to video D, while $A(t, s)$ is obtained from the previous step. The whole process is depicted in Fig. 15.5, where more relevant labels are shown in boldface for the sake of clarity.

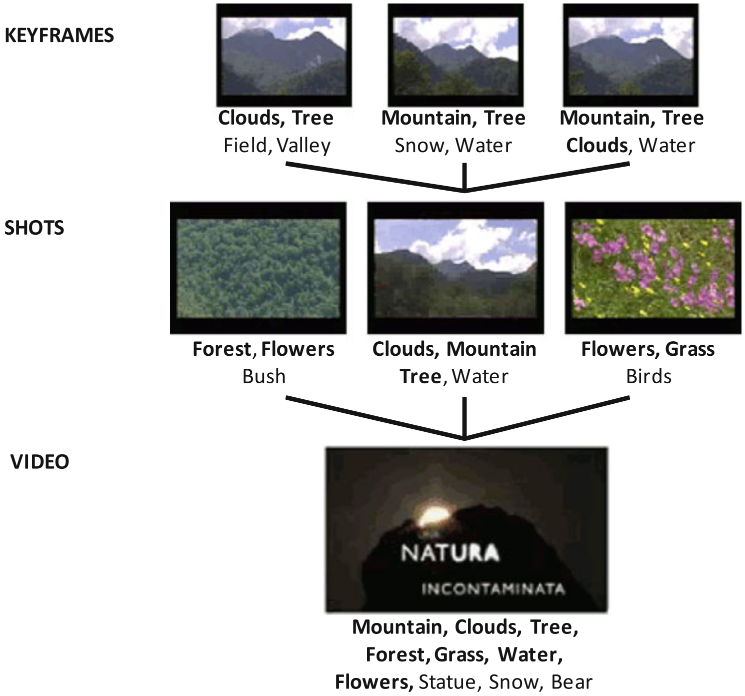

KEYFRAMES
Clouds, Tree Field, Valley
Mountain, Tree Snow, Water
Mountain, Tree **Clouds**, Water
SHOTS
Forest, Flowers Bush
Clouds, Mountain **Tree**, Water
Flowers, Grass Birds
VIDEO
NATURA INCONTAMINATA
Mountain, Clouds, Tree, Forest, Grass, Water, Flowers, Statue, Snow, Bear

Fig. 15.5 Hierarchical annotation: labels suggested for keyframes are summarized into scene and video tags

References

1. Amato, F., Greco, L., Persia, F.: Content-based multimedia information retrieval. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
2. Ardizzoni, S., Bartolini, I., Patella, M.: Windsurf: Region-based image retrieval using wavelets. In: First International Workshop on Similarity Search (IWOSS 1999 - DEXA 1999), vol. 1, pp. 167–173 (1999)
3. Bartolini, I., Ciaccia, P.: Imagination: exploiting link analysis for accurate image annotation. In: Fifth International Workshop on Adaptive Multimedia Retrieval (AMR 2007). Lecture Notes in Computer Science, vol. 4918/2008, pp. 32–44 (2007)
4. Bartolini, I., Ciaccia, P., Waas, F.: Feedbackbypass: a new approach to interactive similarity query processing. In: 27th International Conference on Very Large Data Bases (VLDB 2001), vol. 27, pp. 201–210 (2001)
5. Bartolini, I., Ciaccia, P., Chen, L., Oria, V.: A meta-index to integrate specific indexes: application to multimedia. In: 12th International Conference on Distributed Multimedia Systems (DMS 2006), pp. 29–36 (2006)
6. Bartolini, I., Ciaccia, P., Ntoutsi, I., Patella, M., Theodoridis, Y.: The panda framework for comparing patterns. Data Knowl. Eng. **68**(2), 244–260 (2009)
7. Bartolini, I., Ciaccia, P., Patella, M.: Query processing issues in region-based image databases. Knowl. Inf. Syst. **25**(2), 389–420 (2010)
8. Bartolini, I., Patella, M., Stromei, G.: Efficiently managing multimedia hierarchical data with the windsurf library. In: Obaidat, M.S., Sevillano, J.L., Filipe, J. (eds.) Communications in Computer and Information Science. Lecture Notes in Computer Science, vol. 314, pp. 347–361. Springer, Berlin (2012)
9. Bartolini, I., Patella, M., Romani, C.: Shiatsu: tagging and retrieving videos without worries. Multimedia Tools Appl. J. **63**(2), 357–385 (2013)
10. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Surf: speeded up robust features. Comp. Vis. Image Underst. **110**, 346–359 (2008)
11. Chavez, E., Navarro, G., Baeza-Yates, R., Marroquin, J.L.: Proximity searching in metric spaces. ACM Comput. Surv. **33**, 273–321 (2001)
12. Ciaccia, P., Patella, M., Zezula, P.: M-tree: an efficient access method for similarity search in metric spaces. In: 23rd International Conference on Very Large Data Bases (VLDB'97), Athens (1997)
13. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 102–113, Santa Barbara, CA (2001)
14. Fagin, R., Guha, R., Kumar, R., Novak, J., Sivakumar, D., Tomkins, A.: Multi-structural databases. In: 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Baltimore, MD (2005)
15. Gaede, V., Gunther, O.: Multidimensional access methods. ACM Comput. Surv. **30**, 170–231 (1998)
16. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: 1984 ACM SIGMOD International Conference on Management of Data, Boston, MA, pp. 47–57 (1984)
17. Hearst, M.A.: Clustering versus faceted categories for information exploration. Commun. ACM **49**, 4 (2006)
18. Kleban, J., Moxley, E., Xu, J., Manjunath, B.S.: Global annotation of georeferenced photographs. In: Eighth ACM International Conference on Image and Video Retrieval (CIVR 2009), Article 12, Santorini Island (2009)
19. Lowe, D.G.: Object recognition from local scale-invariant features. In: Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
20. Maron, O., Ratan, A.L.: Multiple-instance learning for natural scene classification. In: 15th International Conference on Machine Learning (ICML 1998), vol. 15, pp. 341–349 (1998)

21. Navigli, R.: Word sense disambiguation: a survey. ACM Comput. Surv. **41**, 2 (2009)
22. Pan, J.Y., Yang, H., Faloutsos, C., Duygulu, P.: Automatic multimedia cross-modal correlation discovery. In: 10th ACM SIGKD International Conference on Knowledge Discovery and Data Mining, vol. 10, pp. 653–658 (2004)
23. Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Trans. Circuits Syst. Video Technol. **8**(5), 644–655 (1998)
24. Wang, L., Khan, L.: Automatic image annotation and retrieval using weighted feature selection. Multimedia Tools Appl. **29**, 55–71 (2006)

Chapter 16
Multimedia Recommendation and Delivery Strategies

Ruggero G. Pensa, Antonio Penta, and Maria Luisa Sapino

16.1 Introduction

In the last decade, the spread of broadband Internet connections even for mobile devices has contributed to an increased availability of multimedia information on the Web. At the same time, due to the decrease of storage cost and the increasing popularity of storage services in the cloud, the problem of information overload has become extremely serious even in personal/company archives. The need to manage, retrieve, and present all these data has promoted the development of advanced multimedia information systems, which include recommendation modules to account for the requests of personalized data selection and presentation.

Recommender systems estimate *ratings*, or *utilities*, which quantify users' degree of interest for the different available data, so that the data can be offered to the users in a personalized way, in decreasing order of interest. Multiple approaches have been proposed in the literature to estimate such degrees of interest. In *content-based filtering* [31], the utility (for a user) of a given item is estimated as a function of the ratings given by the same user to other similar items. For example, in a cultural heritage recommendation application, in order to recommend a monument to a user, content-based filtering relies on the similarity between that monument and the monuments the user has rated highly in the past. (Do they come from the same historical period? Were they designed by the same architect? Do they have the same style? etc.) Then, only the monuments that have a high degree of similarity to the user's preferred ones are recommended. Obviously, the

R.G. Pensa (✉) • A. Penta • M.L. Sapino
University of Torino, C.so Svizzera 185, I-10149 Torino, Italy
e-mail: pensa@di.unito.it; penta@di.unito.it; mlsapino@di.unito.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_16

effectiveness of content-based filtering methods strongly depends on the feature extraction algorithms and on the similarity-based retrieval engine. Content-based techniques only take into account users' past experience and the features of the objects they ranked highly while ignoring the feedback provided by other similar users. A possible drawback of these methods is *overspecialization*, since the system can only recommend items that are similar to those already rated by the user.

A dual approach is *collaborative filtering* [2], in which filtering (i.e., estimating the object's utilities) for a given user is computed by referring to the opinions of other users. Unlike content-based recommendation methods, collaborative systems focus on the similarity among users. Thus, a major challenge faced by collaborative filtering is the need to associate each user with a set of other users having similar profiles: in order to make any recommendations, the system collects data either by asking for explicit users' ratings or through nonintrusive profiling algorithms that implicitly log users' actions. Passive filtering uses aggregates computed on the gathered data (such as the average rating for an item) to make predictions. As a result, each user (similar to the ones whose data have been collected and analyzed) will be given the same predictions for a particular item. Active filtering instead uses patterns in user history to make predictions, thus obtaining user-specific and context-aware useful recommendations. An important limitation of collaborative filtering systems is the so-called cold start problem, i.e., the inability for a recommender to make meaningful recommendations for an object in the absence of ratings by other similar users, thus degrading the filtering performance.

Content-based filtering and collaborative filtering may be manually combined by the end user specifying particular features, essentially constraining recommendations to have certain content features. More often they are automatically combined in the so-called hybrid approach [5, 7, 9, 34] that helps overcome some limitations of each method. Different ways of combining collaborative and content-based methods in a hybrid recommender system can (1) implement collaborative and content-based methods separately and then combine their predictions, (2) incorporate some content-based characteristics into a collaborative approach, (3) incorporate some collaborative characteristics into a content-based approach, and (4) construct a general unifying model that incorporates both content-based and collaborative characteristics. In this chapter, we first present the *co-clustering-based recommendation techniques*, which allow combining heterogeneous multimedia content information and data about the users' preferences and rankings, thus overcoming some of the content-based filtering drawbacks, as well as some collaborative filtering weaknesses. Then, we briefly discuss the challenges in multimedia delivery and the most common strategies adopted in the context of cultural heritage media delivery.

16.2 Grouping of Related Objects and Users Through Co-clustering

In this section, we introduce the co-clustering techniques, which group together related objects and users potentially interested in them.

Each object subject to recommendation may be represented in different and heterogeneous feature spaces. For instance, the picture of a monument may be described by annotations concerning the history of the monument, the materials it has been built with, low-level image features, experts' descriptions, visitors' descriptions and reviews, and so on. Each of these sets of features contributes to the characterization of the objects to different extents. Hence, it is important to consider congruently each type of descriptor during the recommendation process.

"Similar" objects are clustered together, according to a similarity notion that should consider all (or subsets of) the different spaces of features. To this purpose, high-order star-structured co-clustering techniques [14, 18, 20, 22, 29] can be employed to address the problem of heterogeneous data clustering. In this context, the same set of objects is represented in different feature spaces. Such data represent objects of a certain type, connected with other types of data, the features, so that the overall data schema forms a star structure of interrelationships.

The co-clustering task consists of simultaneously clustering the set of objects and the set of values in the different feature spaces. In this way, we obtain a partition of the objects influenced by each of the feature spaces and at the same time a partition of each feature space. Similarly, co-clustering allows to simultaneously group objects and users potentially interested in them.

The recommendation process leverages the clustering results to select a set of candidate objects by using the user's profile, which is modeled as sets of descriptors in the same spaces as the objects' descriptors.

We now provide the formalization of our problem. Let $\mathscr{O} = \{O^1, \dots, O^M\}$ be a set of M multimedia objects and $\mathscr{F} = \{F^1, \dots, F^N\}$ be a set of N feature spaces; a dataset can be viewed under the different views given by the different feature spaces F^k. Therefore, the view k is associated with each feature space F^k. Let $\mathscr{R} = \{R^1, \dots, R^N\}$ be a star-structured relation over $\mathscr{O}$ and $\mathscr{F}$. For each relation matrix R^k, each value $r^k_{st} \in R^k$ corresponds to the counting/frequency/presence of feature $f^k_t \in F^k$ in object $O^s \in \mathscr{O}$. Without loss of generality, we assume that $r^k_{st} \in \mathbb{N}$. An example of two-view star-structured data is given in Fig. 16.1a.

In this recommendation problem, a user is represented as a set of vectors $U = \{\boldsymbol{u}^1, \dots, \boldsymbol{u}^N\}$ in the same N feature spaces describing the objects. Each vector $\boldsymbol{u}^k$ is updated each time the user visits (or revisits) an object, by considering the object features in each space at the instant of the visit. Let $O^U \subset \mathscr{O}$ be the set of objects visited by the user represented by U. Hence, the component of vector $\boldsymbol{u}^k \in U$ related to feature f^k_t is computed as

$$u^k_t = \sum_{O^s \in O^U} r^k_{st}$$

(a)

	f_1^1	f_2^1	f_3^1	f_4^1
O^1	r_{11}^1	r_{12}^1	r_{13}^1	r_{14}^1
O^2	r_{21}^1	r_{22}^1	r_{23}^1	r_{24}^1
O^3	r_{31}^1	r_{32}^1	r_{33}^1	r_{34}^1
O^4	r_{41}^1	r_{42}^1	r_{43}^1	r_{44}^1
O^5	r_{51}^1	r_{52}^1	r_{53}^1	r_{54}^1

	f_1^2	f_2^2	f_3^2
O^1	r_{11}^2	r_{12}^2	r_{13}^2
O^2	r_{21}^2	r_{22}^2	r_{23}^2
O^3	r_{31}^2	r_{32}^2	r_{33}^2
O^4	r_{41}^2	r_{42}^2	r_{43}^2
O^5	r_{51}^2	r_{52}^2	r_{53}^2

(b)

	Y_1^1	Y_2^1	
X_1	t_{11}^1	t_{12}^1	p_1^1
X_2	t_{21}^1	t_{22}^1	p_2^1
	q_1^1	q_2^1	

	Y_1^2	Y_2^2	
X_1	t_{11}^2	t_{12}^2	p_1^2
X_2	t_{21}^2	t_{22}^2	p_2^2
	q_1^2	q_2^2	

Fig. 16.1 An example of a star-structured relation consisting of two feature spaces F^1 and F^2 (**a**) and the contingency tables associated with a related star-structured co-clustering (X, Y^1) and (X, Y^2) (**b**). Each t_{ij}^k represents the contingency value of co-cluster denoted by i and j, p_i^k are the marginals for row clusters denoted by i, and q_j^k are the marginals for column clusters denoted by j

Clearly, the action of updating the vectors in U can be performed incrementally, as the user visits new objects. Notice that, thanks to this approach, users are not described by sets of objects but by sets of features that characterize the objects they visit, like, or browse.

The first step consists of identifying clusters of similar objects in $\mathscr{O}$ by leveraging all feature spaces by means of a star-structured data co-clustering approach. Its goal is to find a set of partitions $\mathscr{Y} = \{Y^1, \ldots, Y^N\}$ over the feature set $\mathscr{F} = \{F^1, \ldots, F^N\}$ and a partition X of the object set $\mathscr{O}$ by optimizing a certain objective function. To solve the high-order star-structured co-clustering problem, several algorithms have been proposed based on different approaches.

For relations involving the set of objects and a unique feature space (such as document-word data), many co-clustering approaches have been proposed. Co-clustering has been studied in many different application contexts including text mining [17], gene expression analysis [16, 32], and graph mining [13] where these methods have yielded an impressive improvement in performance over traditional clustering techniques. The methods differ primarily by the criterion they optimize, such as minimum loss in mutual information [17], sum-squared distance [16], minimum description length (MDL) [13], Bregman divergence [6], and nonparametric association measures [22, 35]. Among these approaches, only the ones based on MDL and association measure are claimed to be parameter-free [25]. However, methods based on MDL are strongly restricted by the fact that they can only handle binary matrices. Association measures, such as Goodman and Kruskal τ, are internal measures of the quality of a co-clustering based on statistical considerations. They also have another advantage: They can deal with both binary and counting/frequency data [22, 35]. From an algorithmic point of view, the co-clustering problem has been shown to be NP-hard [4] when a number of row and column clusters are fixed. Therefore, the methods proposed so far are based on heuristic approaches.

Star-structured (co-) clustering, often referred to as high-order heterogeneous star-structured (co-) clustering, is an emerging topic whose importance is attested by an increasing number of works. Notice also that, in the machine learning community, this field of research is sometimes defined as multiview clustering. Since the topic is quite new, there is no classification for the proposed techniques. However, the existing approaches may be grouped into four main classes: *factorization-based approaches*, *information-theoretic approaches*, *probabilistic approaches*, and *association-based approaches*.

16.2.1 Factorization-Based Approaches

Long et al. [29] use factorization to iteratively embed each type of data object into low-dimensional spaces in a way that takes advantage of the interactions among the different feature spaces. A partitional clustering approach (e.g., k-means) is then employed to obtain the final clustering computed on the transformed spaces. In the above-formulated problem, the approach in [29] tries to minimize

$$L = \sum_{k=1\ldots N} w^k \|R^k - C^O A^k (C^k)^T\|^2$$

where $C^O \in \{0,1\}^{M\times m}$ is a cluster indicator matrix for $\mathcal{O}$ such that $c^O_{pq} = 1$ denotes that pth object in $\mathcal{O}$ is associated with the qth cluster in X. Similarly $C^k \in \{0,1\}^{|F^k|\times n_k}$ is the cluster indicator matrix for Y^k. $A^k \in \mathbb{R}^{m\times n_k}$ is the cluster association matrix such that A^k_{pq} denotes the association between cluster p of X and cluster q of Y^k. Finally, $w^k \in \mathbb{R}_+$ is a weight associated to the kth relation. To compute the clustering X and $\mathcal{Y}$, the proposed algorithm first computes matrices C^O and C^k ($k = 1\ldots N$) by solving a matrix factorization problem. Then, it uses k-means to transform each matrix into an indicator matrix. This method is quite difficult to adopt in practice, since it requires too many parameters: the number of clusters for the object set (m) and for each feature space (n_k, $k = 1\ldots N$) and the weights w^k ($k = 1\ldots N$).

Chen et al. [14] also propose a factorization method that performs multiview co-clustering. The method is an extension of the nonnegative matrix factorization approach that deals with multiview data. The authors formulate the task as an optimization problem with nonnegative matrix trifactorization of $\mathscr{R} = \{R^1, \dots, R^N\}$:

$$J = \min_{G^O \geq 0, G^k \geq 0, S^k \geq 0} \sum_{k=1}^{N} \|R^k - G^O S^k G^k\|^2$$

where $G^O \in \mathbb{R}^{M \times m}$, $G^k \in \mathbb{R}^{|F_k| \times n_k}$ $(k = 1 \dots N)$ are the cluster indicator matrices and $S^k \in \mathbb{R}^{m \times n_k}$ is the cluster association matrix providing the relation between the clusters of objects and the clusters of each feature space. The factorization algorithm consists of an expectation-maximization approach that iteratively updates matrices G^O, G^k, and S^k $(k = 1 \dots N)$.

Additionally, the approach computes new word document and document-category matrices by incorporating user-provided constraints through simultaneous distance metric learning and modality selection. This method is shown to be effective, but its formulation is not flexible. In fact, the number of clusters for each feature space is given as a parameter. Furthermore, the number of parameters grows with the number of feature spaces.

16.2.2 Information-Theoretic Approaches

The information-theoretic co-clustering problem on star-structured data was first considered in [18] where Gao et al. propose to adapt the information theory co-clustering approach [17] to star-structured data. It consists of optimizing a weighted combination of mutual information evaluated over each feature space, where weights are chosen based on the supposed reliability/relevance of their correlation.

In the information-theoretic approaches, partitions X and Y^k $(k = 1 \dots N)$ are defined as discrete random variables. Each variable $Y^k \in \mathscr{Y}$ has n_k categories $Y^k_1, \dots, Y^k_{n_k}$, corresponding to n_k feature clusters, with probabilities $q^k_1, \dots, q^k_{n_k}$, and X has m categories $X_1, \dots, X_m$ corresponding to m object clusters. However, for each variable Y^k, the m categories of X have different probabilities $p^k_1, \dots, p^k_m$, $k = 1 \cdots N$. Probabilities p^k_i and q^k_j are computed as follows:

$$p^k_i = \frac{\sum_{O^s \in X_i} \sum_t r^k_{st}}{\sum_s \sum_t r^k_{st}}, \qquad q^k_j = \frac{\sum_{f^k_t \in Y^k_j} \sum_s r^k_{st}}{\sum_s \sum_t r^k_{st}}$$

The joint probabilities between X and any $Y^k \in \mathscr{Y}$ are denoted by t^k_{ij}, for $i = 1 \cdots m$ and $j = 1 \cdots n_k$, and are computed as follows:

$$t^k_{ij} = \frac{\sum_{O^s \in X_i} \sum_{f^k_t \in Y^k_s} t^k_{st}}{\sum_s \sum_t t^k_{st}}$$

Figure 16.1b provides an example of co-clustering computed on the two-space star-structured data depicted in Fig. 16.1a.

Following [29], the optimal information-theoretic star-structured co-clustering is the one that minimizes

$$D = \sum_{k=1}^{N} \alpha_k \left(I(\hat{Y}^k, \hat{X}) - I(Y^k, X) \right)$$

where $I(\hat{Y}^k, \hat{X}) = \sum_i \sum_j r_{ij}^k \log(\frac{r_{ij}^k}{p_i^k q_j^k})$ is the mutual information, $\hat{X}$ and $\hat{Y}^k$ are partitions where each cluster contains exactly one object/feature, $\alpha_k \geq 0 \ \forall k$, and $\sum_k \alpha_k = 1$.

The optimization approach is an adaptation of the Information Theoretic Co-Clustering (ITCC) algorithm [17]. Beyond the parameters inherited from the original algorithm, the weight α_k involved in the linear combination also has to be fixed by the end user. Another drawback of this approach is its complexity that prevents its use on large-scale datasets. Greco et al. [20] propose a similar approach based on the linear combination of mutual information evaluated on each feature space, where the parameter of the linear combination is automatically determined.

16.2.3 Probabilistic Approaches

In [30], a parametric probabilistic approach to cluster relational data is proposed. A Monte Carlo simulation method is used to learn the parameters and to assign objects to clusters. The problem of clustering images described by segments and captions is considered in [10]. The proposed algorithm is based on Markov random fields in which some of the nodes are random variables in the combinatorial problem. Ramage et al. [33] propose a generative clustering algorithm based on latent Dirichlet allocation to cluster documents using two different sources of information: document text and tags. Each source is modeled by a probability distribution, and a weight value is used to weigh one vector space with respect to the other. During the learning step, the algorithm finds the distribution parameters and models documents, words, and tags. In addition to the weight parameter, the method has another drawback: it constrains the number of hidden topics in text and tag sources to be the same, which is a strong assumption on data that is not always true.

16.2.4 Association-Based Approaches

In [22], Ienco et al. present a parameter-less iterative algorithm that maximizes the Goodman-Kruskal τ, a statistical measure of association that automatically identifies a congruent number of high-quality co-clusters. We provide in-depth information on this approach because it is parameter-less, i.e., contrary to the other approaches, it does not require a user-defined number of clusters. Goodman and Kruskal τ measure [19] estimates the association between two categorical variables X and Y by the proportional reduction of the error in predicting X knowing or not the variable Y:

$$\tau_{X|Y} = \frac{e_X - E[e_{X|Y}]}{e_X}$$

Evaluating the quality of the partition of objects, given the partitions of features, is formalized as follows. The partition of objects is considered as the dependent variable X, and the N partitions of the feature spaces are considered as many independent variables $\mathscr{Y} = \{Y^1, \dots, Y^N\}$. X and Y are defined as for the information-theoretic co-clustering setting.

The error in predicting X is the sum of the errors over the independent variables of $\mathscr{Y}$: $e_X = \sum_{k=1}^{N} \sum_{i=1}^{m} p_i^k(1-p_i^k) = N - \sum_{k=1}^{N} \sum_{i=1}^{m} (p_i^k)^2$. $E[e_{X|\mathscr{Y}}]$ is the expectation of the conditional error taken with respect to the distributions of all $Y^k \in \mathscr{Y}$:

$$E[e_{X|\mathscr{Y}}] = \sum_{k}^{N} \sum_{j}^{n_k} q_j^k \, e_{X|Y_j^k} = \sum_{k}^{N} \sum_{j}^{n_k} q_j^k \sum_{i}^{m} \frac{t_{ij}^k}{q_j^k}\left(1 - \frac{t_{ij}^k}{q_j^k}\right) = N - \sum_{k}^{N} \sum_{i}^{m} \sum_{j}^{n_k} \frac{(t^k{}_{ij})^2}{q_j^k}$$

The generalized Goodman-Kruskal's $\tau_{X|\mathscr{Y}}$ association measure is then equal to

$$\tau_{X|\mathscr{Y}} = \frac{e_X - E[e_{X|\mathscr{Y}}]}{e_X} = \frac{\sum_k \sum_i \sum_j \frac{(t^k{}_{ij})^2}{q_j^k} - \sum_k \sum_i (p_i^k)^2}{N - \sum_k \sum_i (p_i^k)^2} \tag{16.1}$$

If we consider Y^k as a dependent variable and X as an independent variable, the corresponding $\tau_{Y^k|X}$ is computed as follows:

$$\tau_{Y^k|X} = \frac{e_{Y^k} - E[e_{Y^k|X}]}{e_{Y^k}} = \frac{\sum_i \sum_j \frac{(t_{ij}^k)^2}{p_i^k} - \sum_j (q_j^k)^2}{1 - \sum_j (q_j^k)^2} \tag{16.2}$$

The adopted co-clustering approach for star-structured data is formulated as a multi-objective combinatorial optimization problem which aims at optimizing $N+1$ objective functions based on Goodman-Kruskal's τ measure. The main procedure of the algorithm is shown in Fig. 16.2. The reader may refer to [22] for further algorithmic details.

To provide a first candidate list of objects to be recommended, one can measure the *cosine similarity* of each user vector associated to the kth space, with the

```
Input: a star-structured dataset $\mathcal{SD}$ and an integer $N_{iter}$
Output: a coclustering $(X, \mathcal{Y})$
  Initialize $Y^1, \cdots, Y^N$, $X$ with discrete partitions
  $i \leftarrow 0$
  $T \leftarrow \emptyset$
  for $k = 1$ to $N$ do
    $T^k \leftarrow$ CONTINGENCYTABLE$(X, Y^k, SD^k)$
    $T \leftarrow T \bigcup T^k$
  end for
  while $(i \leq N_{iter})$ do
    $[X, T] \leftarrow$ OPTIMIZEMULTIOBJECTCLUSTER$(X, \mathcal{Y}, T)$
    for $k = 1$ to $N$ do
      $[Y^k, T^k] \leftarrow$ OPTIMIZEFEATURECLUSTER$(X, Y^k, T^k)$
    end for
    $i \leftarrow i + 1$
  end while
  return $Y^1, \cdots, Y^N, X$
```

Fig. 16.2 Pseudo code of the adopted star-structured co-clustering algorithm [22]

centroids of each object cluster in the kth space. Let $\boldsymbol{x}_i^k$ be the centroid of cluster X_i in the feature space F^k. The tth component of $\boldsymbol{x}_i^k$ is computed as

$$x_i^k = \frac{\sum_{O^s \in X_i} d_{st}^k}{|X_i|}$$

and the cosine similarity between $\boldsymbol{u}^k$ and $\boldsymbol{x}_i^k$ is evaluated as

$$\mathrm{sim}(\boldsymbol{u}^k, \boldsymbol{x}_i^k) = \frac{\boldsymbol{u}^k \cdot \boldsymbol{x}_i^k}{\|\boldsymbol{u}^k\| \, \|\boldsymbol{x}_i^k\|}$$

For each space, the most similar object cluster is chosen, leading to a set of N clusters $\mathcal{X}^c = \{X_1^c, \ldots, X_N^c\}$ of candidate objects. Then, two different strategies can be adopted to provide the prefiltered list of candidate objects $\mathcal{O}^c$:

- **Relaxed strategy**: the objects belonging to the union of all clusters are retained, i.e.,

$$\mathcal{O}^c = \bigcup_k X_k^c$$

- **Strict strategy**: the most represented cluster in $\mathcal{X}^c$ is retained, i.e.,

$$\mathcal{O}^c = \underset{X_k^c \in \mathcal{X}^c}{\mathrm{argmax}} \left| X_l^c \in \mathcal{X}^c \;\; s.t. \;\; X_k^c \equiv X_l^c \right|$$

The first strategy is suitable when the user's vectors are associated to very small clusters (e.g., because the user likes very uncommon objects). In any other situation, the second strategy is the most appropriate. As an additional step, objects already visited/liked/browsed by the user can be filtered out. We do not filter out these objects at the beginning of the prefiltering stage because they are relevant for the co-clustering step. In fact they are likely to be involved in important cross-associations between sets of features and sets of objects.

Finally, provided that each object in $\mathscr{O}$ is geo-referenced, the set of candidate objects $\mathscr{O}^c$ issued by the above-described process can be further refined by an ordering step. To this purpose, we employ the route distance between the user's current position and the position of each object in $\mathscr{O}^c$. Closer objects are on top of the items' list, while more distant ones are at the bottom. In conclusion, at the end of the prefiltering stage, we provide an ordered list of candidate objects $\widehat{\mathscr{O}^c}$ grouped by the related cultural points of interest (POI) (in this manner a user can easily choose items coming from more different cultural POIs).

16.3 Delivery Strategies for Multimedia Recommendation

Content-based filtering, collaborative filtering, and co-clustering-based recommendation techniques described in the previous section help in identifying the multimedia items a user might be interested in. Suitable delivery strategies are then applied to deliver the identified multimedia objects (ranked in decreasing order of expected interest for the user), in order to fit best the user's requests. Contents are adapted to the user's request relying on contextual information, such as the location the users are in, the device they are using to retrieve the information of interest, their profile, and their search history. Delivery strategies can overcome several drawbacks of common approaches of classical multimedia recommendation systems. In fact, although the users' requirements are often expressed in terms of high-level descriptions of the desired contents, it is not always possible to automatically extract meaningful high-level information from multimedia features and directly use such features in the recommendation algorithms. Thus, using context information can help increase the performance of recommendation systems by filtering out those items that do not match the user needs in the given context. Also, for some kinds of multimedia data, there does not exist a precise correlation between high- and low-level features (e.g., in images the concept of "moon" is related to a region with a circular shape and white color with a given uncertainty). It is important to understand the semantics of the users' query and rely on it to strengthen or weaken the ranking of the objects identified as interesting by the content-based recommendation systems.

Users' preferences and feedbacks are not always explicitly known and available. This is especially the case when the multimedia system does not require a registration—with the specification of profiling information—from the users. Contextual information can help in estimating users' preferences at the query time, maybe also taking into account the features of the objects the user is currently observing. For example, the main colors of the painting the user is watching can give hints on the corresponding artistic movement or school and can be taken into account when identifying other paintings to suggest. In the next subsections, we will provide a brief survey of the most common approaches used to define the delivery strategies.

16.3.1 Context-Based Delivery Strategies

There are many definitions of what is a context. Abowd et al. [1] define the context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant for the interaction between a user and an application, including the user and the application themselves."

In a recent survey [11], different context modeling approaches, including object-role-based models, spatial models, and ontology-based models, are presented, together with a discussion on how context-based reasoning can enhance the quality of many services.

Many multimedia recommendation platforms use context information for delivering multimedia services. For example, in [41] the authors describe a system that can handle many context categories like user preferences, situation, and capability context. Their model takes into account an ontology which delineates attributes like user situation and user media preferences using namespaces and concepts derived from MPEG-7 descriptors of the media metadata. Recently, in [8] a multimedia recommendation system was presented that extends the classical recommendation delivery strategies by supporting useful *context-aware* services (e.g., a multimedia touristic guide). Such services are developed to assist users when visiting cultural environments (indoor museums, archeological sites, old town centers) containing several *cultural POIs* (e.g., paintings of museum rooms, buildings in ancient ruins or in an old town center, etc.) correlated with a large amount of multimedia data available in multiple Web repositories. It has been shown in real case studies in both outdoor and indoor scenarios that this approach is successful, in terms of both users' satisfaction and system accuracy. Among the hybrid solutions, the uMender system [39] exploits context information, musical content, and relevant user ratings to perform music recommendations on mobile devices. A framework for recommendation of multimedia objects based on processing of individual ontologies with context information is proposed in [23]: the recommendation process takes into account similarities calculated both between objects' (metadata) and users' ontologies, which reflect the social and semantic features managed by the system. Smart TV systems are multimedia systems which easily present different types of multimedia content to end users. Recently, some of those systems have also developed personalization techniques to recommend the most suitable content to users, exploring approaches from content-based user modeling to group-based collaboration. Lee et al. [28] describe a smart TV system with several unique features such as a Kinect-based component to recognize human body gestures for TV control, social tags, and various environmental situations to annotate multimedia items and improve users' recommendation accordingly. In [40], a useful taxonomy for mobile multimedia recommender systems has been presented, which is based on context-aware services. The classification of those systems is based on the type of collected information (explicit or implicit feedback), the type of recommendation learning process, and the algorithms used to make prediction/recommendation.

16.3.2 Location-Based Delivery Strategies

Due to the recent increase in the availability of powerful mobile devices, location-aware systems are becoming more widespread, and recommendation systems are used to find interesting events, places, and objects that are close to users' locations. Bobadilla et al. [12] provide a survey of different location-based recommendation systems. An example showing the need for location-aware delivery systems is the case in which users rate cultural POIs using multiple different features, not including the distance at the time of voting between the users and the point of interest (POI) they are voting. However, users who rely on mobile personalized touristic guides expect that the ranking in the recommendation of a point of interest takes into account not only the good ratings from similar users but also the distance between their current position and the considered point of interest. Location awareness might imply a reordering of the rankings of a set of interesting points in such a way that a driving path reaching all the potentially interesting nearby points can be identified. In the context of multimedia data recommendation, [24] presents a system to recommend music well suited for POIs. The considered scenario consists of a mobile city guide based on an enhanced presentation of places of interest for tourists, in which music related to each point of interest being described (i.e., music that is culturally or emotionally associated with the place) is played. Similar systems are presented in [15], where the proposed models take into account the influence of online music social trends on users' local preferences. Schedl et al. [38] describe a geospatial model taking into consideration GPS coordinates and semantic locations (continent, country, and state) of the user. In [42], a recommender system is described that correlates viewable scene information from sensors with geographic contextual tags from OpenStreetMap. The co-occurrence of geo-tags and mood tags is computed based on a set of categories of the web site "Foursquare.com," and a mapping from geo-tags to mood tags is obtained. The music retrieval component returns music based on matching mood tags.

A special case of location-based recommendation is the one referring to the concept of "smart space," as defined in [37]. In this case, multimedia systems that deliver recommended content as the interactive TV applications encapsulate both the information in a physical space and the information about the access to this information. This kind of location becomes a dynamic environment that changes over time, reflecting the way the different entities interact with it to share information among them.

16.3.3 Delivery Strategies Based on Device Features

In the definition of the delivery strategy for the recommendation algorithms, the effect of the device in which the selected media will be actually played has great importance. In fact, users access multimedia systems using different devices, including desktop or laptop computers, smartphones, tablets, etc. Each device has its own interface characteristics (e.g., display capability); its specific Internet connection parameters, including cost and upload/download speed; and different storage space and computational capabilities. These differences have an impact on user behavior and preferences; for example, when we use a cellular phone, we could prefer to download a lighter multimedia content than when we use a cabled device. Thus, multimedia content delivery should be adapted to the different devices. To face this problem, Rosaci et al. in [36] have proposed a multimedia Web service whose architecture allows to compute multi-device context-aware recommendations using an agent-based system. On the other hand, it is possible to introduce device adaptation features in the specification of multimedia documents to support the delivery of different versions of the same document on different devices, taking into account the device characteristics. In this line, in [27] the authors describe a framework for standard multimedia documents based on an abstract structure that captures the spatiotemporal and hypermedia dimensions of multimedia documents and propose an algorithm which transforms (in a minimal way) such multimedia documents to satisfy the presentation device constraints.

16.3.4 Profile-Based Delivery Strategies

One of the classical modules in recommendation systems is the user profiling module, which learns (or at least estimates) users' interests over a long period of time by analyzing users' history, their inputs, and/or their relationships . Most state-of-the-art user profiling approaches are based on the textual content of relevant documents to identify these interests. Hopfgartner et al. in [21] exploited the Linked Open Data Cloud to identify similar news stories that match the users' interest to support the intelligent delivering of multimedia news. Albanese et al. in [3] described a multimedia recommendation system which combines the intrinsic features of multimedia objects, past behavior of individual users, and overall behavior of the entire community of users resembling the well-known PageRank ranking strategy. Konstas et al. in [26] used the additional relationships in a social network as user profile to develop a track recommendation system, thus taking into account both the social annotation and friendships inherent in the social graph established among users, items, and tags in order to create a collaborative recommendation system that effectively adapts to the personal information needs of each user.

16.4 Conclusion

The need to manage, retrieve, and present multimedia information on the Web has promoted the development of advanced multimedia information systems, which include recommendation modules to account for the requests of personalized data selection and presentation. Multiple approaches have been proposed in the literature to estimate users' degree of interest for the different available data. In this chapter, we have presented *co-clustering-based recommendation techniques*, which allow to combine heterogeneous multimedia content information and data about the users' preferences and rankings, thus overcoming some of the content-based filtering drawbacks, as well as some collaborative filtering weaknesses. Then, we briefly discussed the challenges in multimedia delivery and the most common strategies adopted in the context of cultural heritage media delivery.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, HUC '99, pp. 304–307. Springer, Berlin (1999)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(2), 734–749 (2005)
3. Albanese, M., d'Acierno, A., Moscato, V., Persia, F., Picariello, A.: A multimedia recommender system. ACM Trans. Internet Technol. **13**(1) (2013)
4. Anagnostopoulos, A., Dasgupta, A., Kumar, R.: Approximation algorithms for co-clustering. In: Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '08, pp. 201–210. ACM, New York, NY (2008)
5. Anand, S.S., Kearney, P., Shapcott, M.: Generating semantically enriched user profiles for web personalization. ACM Trans. Internet Technol. **7**(4), 22 pp, (2007)
6. Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., Modha, D.S.: A generalized maximum entropy approach to bregman co-clustering and matrix approximation. J. Mach. Learn. Res. **8**, 1919–1986 (2007)
7. Bartolini, I., Zhang, Z., Papadias, D.: Collaborative filtering with personalized skylines. IEEE Trans. Knowl. Data Eng. **23**(2), 190–203 (2011)
8. Bartolini, I., Moscato, V., Pensa, R., Penta, A., Picariello, A., Sansone, C., Sapino, M.: Recommending multimedia visiting paths in cultural heritage applications. Multimedia Tools Appl. J., 1–30 (2014)
9. Basilico, J., Hofmann, T.: Unifying collaborative and content-based filtering. In: Proceedings of the 21st International Conference on Machine Learning (ICML 2004), Banff 4–8 July 2004

10. Bekkerman, R., Jeon, J.: Multi-modal clustering for multimedia collections. In: Computer Vision and Pattern Recognition, IEEE Computer Society Conference on Vision and Pattern Recognition, pp. 1–8 (2007)
11. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010). Context Modelling, Reasoning and Management
12. Bobadilla, J., Ortega, F., Hernando, A., Gutiarrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**(0), 109–132 (2013)
13. Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully automatic cross-associations. In: KDD, pp. 79–88 (2004)
14. Chen, Y., Wang, L., Dong, M.: Semi-supervised document clustering with simultaneous text representation and categorization. In: ECML/PKDD (1), pp. 211–226 (2009)
15. Cheng, Z., Shen, J.: Just-for-me: an adaptive personalization system for location-aware social music recommendation. In: Proceedings of ACM International Conference on Multimedia Retrieval (ICMR 2014), Glasgow, April 2014
16. Cho, H., Dhillon, I.S., Guan, Y., Sra, S.: Minimum sum-squared residue co-clustering of gene expression data. In: Proceedings of SIAM SDM 2004 (2004)
17. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of ACM SIGKDD 2003, pp. 89–98. ACM Press, Washington (2003)
18. Gao, B., Liu, T.Y., Ma, W.Y.: Star-structured high-order heterogeneous data co-clustering based on consistent information theory. In: ICDM '06: Proceedings of the Sixth International Conference on Data Mining, pp. 880–884 (2006)
19. Goodman, L.A., Kruskal, W.H.: Measures of association for cross classification. J. Am. Stat. Assoc. **49**, 732–764 (1954)
20. Greco, G., Guzzo, A., Pontieri, L.: Co-clustering multiple heterogeneous domains: linear combinations and agreements. IEEE Trans. Knowl. Data Eng. **22**(12), 1649–1663 (2010)
21. Hopfgartner, F., Jose, J.M.: Semantic user profiling techniques for personalised multimedia recommendation. Multimedia Syst. **16**(4–5), 255–274 (2010)
22. Ienco, D., Pensa, R.G., Meo, R.: Parameter-free hierarchical co-clustering by n-ary splits. In: Proceedings of ECML/PKDD 2009. Lecture Notes in Computer Science, vol. 5781, pp. 580–595. Springer, Berlin (2009)
23. Juszczyszyn, K., Kazienko, P., Musiał, K.: Personalized ontology-based recommender systems for multimedia objects. In: Hãkansson, A., Hartung, R., Nguyen, N.T. (eds.) Agent and Multi-agent Technology for Internet and Enterprise Systems. Studies in Computational Intelligence, pp. 275–292. Springer, Berlin (2010)
24. Kaminskas, M., Ricci, F., Schedl, M.: Location-aware music recommendation using auto-tagging and hybrid matching. In: Proceedings of the 7th ACM Conference on Recommender Systems (RecSys 2013), Hong Kong, 12–16 October 2013
25. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards parameter-free data mining. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, pp. 206–215. ACM, New York, NY (2004)
26. Konstas, I., Stathopoulos, V., Jose, J.M.: On social networks and collaborative recommendation. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, pp. 195–202. ACM (2009)
27. Laborie, S., Euzenat, J., Layaïda, N.: Semantic adaptation of multimedia documents. Multimedia Tools Appl. **55**(3), 379–398 (2011)
28. Lee, W.P., Kaoli, C., Huang, J.Y.: A smart {TV} system with body-gesture control, tag-based rating and context-aware recommendation. Knowl.-Based Syst. **56**(0), 167–178 (2014)
29. Long, B., Zhang, Z.M., Wú, X., Yu, P.S.: Spectral clustering for multi-type relational data. In: ICML '06: Proceedings of the 23rd International Conference on Machine Learning, pp. 585–592 (2006)
30. Long, B., Zhang, Z.M., Yu, P.S.: A probabilistic framework for relational clustering. In: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–479 (2007)

31. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. Springer, Berlin (2007)
32. Pensa, R.G., Boulicaut, J.F.: Constrained co-clustering of gene expression data. In: Proceedings of SIAM SDM 2008, pp. 25–36 (2008)
33. Ramage, D., Heymann, P., Manning, C.D., Garcia-Molina, H.: Clustering the tagged web. In: WSDM, pp. 54–63 (2009)
34. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook. Springer, Berlin (2011)
35. Robardet, C., Feschet, F.: Comparison of three objective functions for conceptual clustering. In: Proceedings PKDD'01. Lecture Notes in Artificial Intelligence, vol. 2168, pp. 399–410. Springer, Berlin (2001)
36. Rosaci, D., Sarno, G.: Recommending multimedia web services in a multi-device environment. Inf. Syst. **38**(2), 198–212 (2013)
37. Saleemi, M., Lilius, J.: Exploiting smart spaces for interactive tv applications development. J. Supercomput., **30**(3), 1200–1217 (2014)
38. Schedl, M., Vall, A., Farrahi, K.: User geospatial context for music recommendation in microblogs. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014), Queensland, July 2014
39. Su, J.H., Yeh, H.H., Yu, P.S., Tseng, V.S.: Music recommendation using content and context information mining. IEEE Intell. Syst. **25**(1), 16–26 (2010)
40. Xia, F., Asabere, N., Ahmed, A., Li, J., Kong, X.: Mobile multimedia recommendation in smart communities: a survey. CoRR **1**, 606–624 (2013)
41. Yu, Z., Zhou, X., Zhang, D., Chin, C.Y., Wang, X., Men, J.: Supporting context-aware media recommendations for smart phones. IEEE Pervasive Comput. **5**(3), 68–75 (2006)
42. Yu, Y., Shen, Z., Zimmermann, R.: Automatic music soundtrack generation for outdoor videos from contextual sensor information. In: ACM Multimedia, pp. 1377–1378 (2012)

Part VI
Application to the DATABENC Case Study

In this final part, we show an application of discussed data management pervasive technologies for a cultural heritage smart scenario by describing the PATCH (*Portable context-aware ATlas for Cultural Heritage*) system—a "portable" prototype of a multimedia and social "atlas" of cultural points of interest, which browsing is driven by the context.

The system is characterized by several features typical of pervasive systems: (a) capability of gathering information from distributed and heterogeneous pervasive data sources (e.g., sensor networks, social networks, digital libraries and archives, multimedia collections, Web data services, etc.); (b) context awareness and, consequently, ability to provide useful and personalized data and services for users on the base of their preferences and of the surrounded environment; (c) advanced data management techniques and technologies necessary to deal with the information variety, velocity, and volume; and (d) advanced smart services as retrieval, recommendation, analytics, and other utilities.

A specific application of our system within DATABENC is proposed by means of a couple of real case studies related to the *Paestum Ruins* and to Salerno old town.

Chapter 17
PATCH: A Portable Context-Aware ATlas for Browsing Cultural Heritage

Francesco Colace, Massimo De Santo, Vincenzo Moscato, Antonio Picariello, Fabio A. Schreiber, and Letizia Tanca

17.1 Introduction

Offering environments for virtual navigation of cultural items turns out to be particularly important for the valorization and promotion of worldwide *cultural heritage*. This need is particularly perceived in a country like Italy, whose artistic patrimony embodies a universal resource of inestimable value, attracting millions of visitors every year to monuments, archaeological sites, and museums.

Embedding this patrimony within a pervasive digital ecosystem provides the final users with very useful value-added services and information coming from rich and heterogeneous sources while allowing for the collection of data that can then be profitably analyzed offline (see Chap. 1 [16]).

In this new scenario, we can imagine *persons* (citizens, tourists, etc.) and *objects* (items, buildings, rooms, etc.) equipped with appropriate devices (GPS, smartphone, video cameras, temperature/humidity sensors, etc.) as the building blocks of a *pervasive environment* in which all the mentioned entities, and possibly many more, communicate and operate. The data produced and exchanged can be opportunely exploited by applications that contribute to making the system "smart."

Indeed, the system is grounded in a set of *cultural points of interests*, C_{PoI} (indoor museums, archaeological sites, old town center, etc., but also a single sculpture,

F. Colace (✉) • M. De Santo
University of Salerno, Salerno, Italy
e-mail: fcolace@unisa.it; desanto@unisa.it

V. Moscato • A. Picariello
University of Naples Federico II, Naples, Italy
e-mail: vmoscato@unina.it; picus@unina.it

F.A. Schreiber • L. Tanca
Politecnico di Milano, Milano, Italy
e-mail: fabio.schreiber@polimi.it; letizia.tanca@polimi.it

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0_17

a picture within a museum, etc.), each needing particular ICT infrastructures and services that transform the physical spaces into smart environments.

In this final chapter, we show the application of the pervasive technologies introduced in the previous chapters to a cultural heritage smart scenario by describing the **PATCH** (*Portable context-aware ATlas for Cultural Heritage*) system—a "portable" prototype of a multimedia and social "atlas" of cultural points of interest, for browsing of which is driven by the context.

The system—developed within DATABENC,[1] the high-technology district for cultural heritage management recently funded by the Campania Region in Italy—is characterized by several features typical of pervasive systems:

- Capability to gather information from distributed and heterogeneous pervasive data sources (e.g., sensor networks, social networks, digital libraries and archives, multimedia collections, Web data services, etc.)
- Context awareness and, consequently, ability to provide useful and personalized data and services to users based on their preferences and on the current situation and surrounding environment
- Advanced data management techniques and technologies necessary to deal with information variety, velocity, and volume
- Advanced smart services such as information retrieval, recommendation, data analytics, and other utilities

In particular, PATCH is a multimedia and social atlas of all the cultural heritage resources in Campania, which can be used to get detailed information about C_{PoI}s in a given geographic area and to access the smart services related to them. This information can include:

- Items' descriptions—coming from open Web sources (e.g., *Wikipedia*) or from cultural digital libraries and archives
- Multimedia data—video, text, image, and audio, likewise associated to the various items of interest, retrieved from open (*Panoramio*, *Picasa*, *YouTube*, *Facebook*, and *Flickr*) or private collections
- Social data—e.g., user comments and opinions from common social networks like *Facebook*, *Twitter*, etc.
- Sensor data—related to some environmental parameters (e.g., humidity and temperature) and captured by proper sensor networks deployed in the C_{PoI}
- Web services data—any kind of useful data gathered by means of Web services, e.g., about other touristic attractions, or accommodations in the same geographic area, or meteorological information

Consistently, the services can comprise:

- Retrieval and presentation functionalities to search some information of interest and present it to the users in a suitable format

[1] www.databenc.it

- Recommendation facilities suggesting cultural visiting paths to users or helping them browse multimedia information as a sort of multimedia guide
- Social sensing, in other terms the possibility of exploring the general mood of a social community and of adding the personal opinions and comments about a C_{PoI}
- Data analytics, reporting and visualizing statistics on a given C_{PoI} (e.g., number of visiting users, present and future environmental and weather conditions, user feedback, etc.)
- Other utilities, such as the possibility to book the visit to an exhibition or a table in a restaurant in the same area, to check for parking lots availability, etc.

The described data and services are dynamically tailored to the single users in a context-driven way.

A further characteristic of PATCH is its *portability*: each user is endowed with, and can carry, a personalized version of the atlas, in which the relevant information dynamically changes with the context. Typically, users consulting PATCH while at home will be presented with information and services that can be different from what they get when physically close to a C_{PoI}. For example, when at home the user will receive general information on how to reach the various C_{PoI}s, along with summarized descriptions of them all, while when at the specific C_{PoI}, another set of data and services (meteorological information, details on art pieces, or booking facilities related to the specific place) would be useful.

The chapter is organized as follows: Sect. 17.2 shows a couple of motivating examples of the proposed system. Section 17.3 describes the system with the related functionalities. Section 17.4 presents some implementation details, and finally, Sect. 17.5 discusses some conclusions and future work.

17.2 Case Studies

In this section, we describe two case studies that have been designed and implemented within the DATABENC project in order to show the main ideas and the related technologies that are beyond PATCH: the famous *Paestum archaeological site* and *Salerno historical district*.

Paestum Archaeological Site—The archaeological site of *Paestum* is one of the major Greco-Roman cities in the South of Italy.

Here, a set of ancient buildings is the main cultural attraction for a tourist: three main temples of Doric style (the *First Temple of Hera*, also called *Basilica*; the *Second Temple of Hera*, also known as *Temple of Neptune*;

(continued)

and the *Temple of Athena*), the *Roman Forum* with several ruins, and the *amphitheater*.

All the buildings are surrounded by the remains of the city walls. In addition, there is a museum near the ancient city containing evidence of the Greco-Roman life (e.g., amphorae, paintings, and other objects).

All the ancient buildings together with the museum will constitute the set of C_{PoI}s for our case study.

Tourists, both from their homes and while visiting the ruins, can browse the atlas and enjoy a useful multimedia guide describing the main cultural attractions of Paestum or be recommended other nearby places with *visiting paths*, comments of other users, and other information of interest. Initially, a user at the ruins has the possibility of visualizing on a map the different C_{PoI}s along with a set of basic information (e.g., name of C_{PoI} and other general metadata, general users' impression, meteorological information, number of visitors actually present in the same area, etc.) and services (e.g., booking of a restaurant near the ruins or of an exhibition within the museum).

When a user approaches a particular C_{PoI} (e.g., Temple of Neptune), or selects it explicitly, a "change of context" occurs, and the related multimedia description (i.e., audio, images, video, and texts) and detailed users' comments are delivered together with the activation of a set of more specific services.

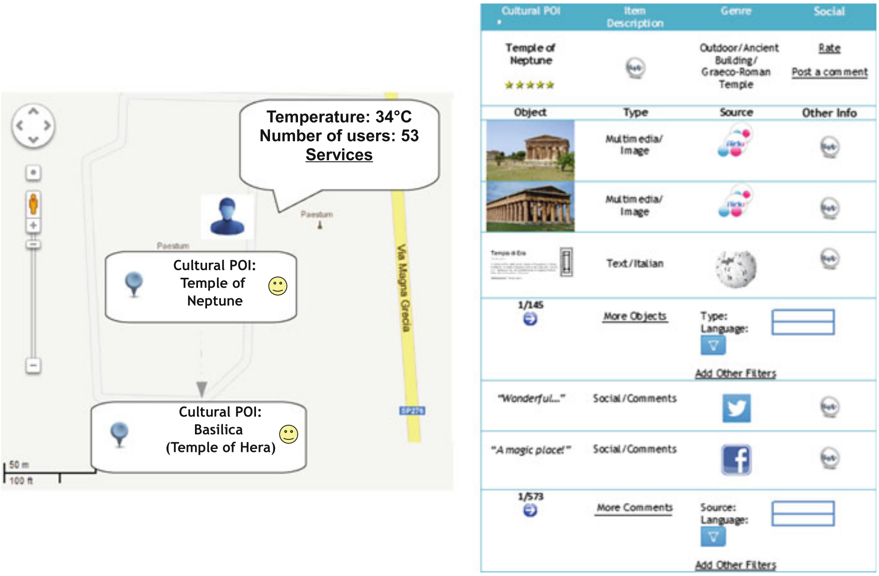
The list of proposed multimedia objects depends on the user's preferences (e.g., images rather than voice or expert-level rather than layman-level descriptions of the art pieces), and their browsing can be facilitated using multimedia recommendation facilities.

At the same time, users can choose to retrieve some interesting information; to read comments, opinions, and ratings about the visited C_{PoI}; and to express their own ratings and opinions that will be posted on social network. Possibly, details about environmental conditions within the temple and other services can be provided together with the suggestion of cultural visiting paths.

When a user approaches a different C_{PoI}, a new list of objects is automatically delivered (e.g., removing those related to the previous visited C_{PoI} and adding the new objects) with a different, more appropriate set of services, and consequently the visiting path will be automatically updated.

The paths can take into account the current context in terms of actual position (obtained in this case by GPS) and the weather and environmental conditions, thus enhancing the visiting experience. This information acquired, the path can dynamically change also due to the presence of temporary restrictions to enter the areas (e.g., too high temperature/humidity or a closed area).

Figure 17.1 shows a running example for our system concerning a visiting path for the Paestum Ruins. Users can browse through the data by means of a proper Graphical User Interface (GUI); she/he can filter objects belonging to a given C_{PoI} using different criteria: type of multimedia data, language, size, etc.



Cultural POI	Item Description	Genre	Social
Temple of Neptune ★★★★★		Outdoor/Ancient Building/ Graeco-Roman Temple	Rate Post a comment
Object	**Type**	**Source**	**Other Info**
	Multimedia/ Image		
	Multimedia/ Image		
	Text/Italian		
1/145	More Objects	Type: Language: Add Other Filters	
"Wonderful..."	Social/Comments		
"A magic place!"	Social/Comments		
1/573	More Comments	Source: Language: Add Other Filters	

Fig. 17.1 PATCH: a running example

> Salerno Historical District—The old town of *Salerno* (a city in the South of Italy) has a great number of historical and cultural assets and combines evidence (e.g., several places, monuments, artworks, and archival documents) coming from several ages: from its foundation as an Osco-Etruscan center and the Roman period up to more recent times with the Swabians, the Angevins, and the Aragonese.

This area represented a challenge for our application, aiming at enhancing the related archaeological, architectural, and cultural heritage: differently from the previous case study related to an old site, the Salerno historical district is a living urban area with all the related constraints.

Here the atlas stores and provides all the information related to the cultural heritage aspects (archaeological sites, monuments, ancient and modern buildings, cultural and landmark assets) in a georeferenced environment, thus providing a huge amount of data that can be analyzed also by analytics systems. In particular, the atlas contains[2]:

- All the information related to a historical district information system
- An archaeological map of the area, one for each considered era

[2] See also the web site `www.tempiconnessi.info` for more info.

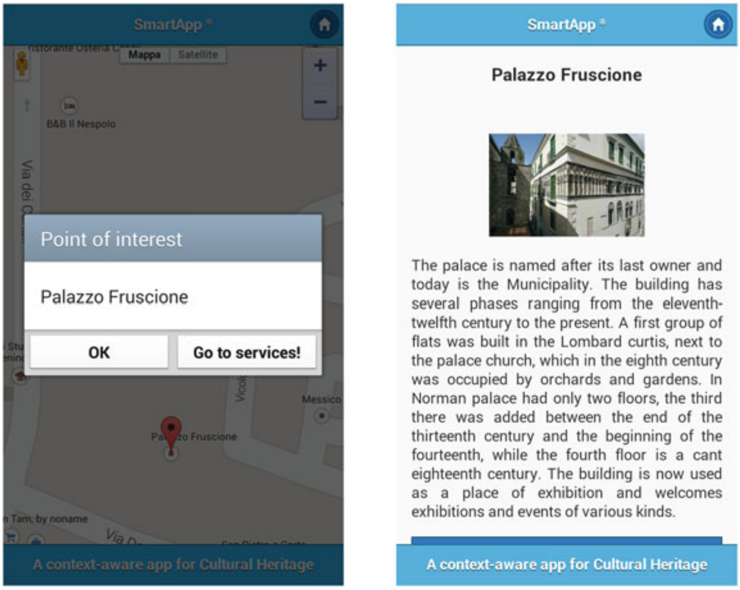
- An analysis of the inner wall structures
- All the information for connecting to a digital library of manuscripts, essay, texts, and so on

In this context, the atlas assists tourists providing visiting paths, emphasizing, on the one hand, the archaeological, architectural, and historical assets of Salerno and, on the other hand, the persistence of the past and the most recent urban transformations.

For example, a tourist walking around *Piazza Portanova* can visualize on the map different C_{PoI}s with basic descriptions, available digital resources, and utility services. Based on the users' current position (GPS) and their profile and other different conditions forming a certain context—for example, the weather conditions—the system suggests an outdoor visiting path through *Via dei Mercanti* or *Piazza del Crocifisso* rather than an indoor visit to the *Medical School Museum* located in the ancient *Church of St. Gregory* (1058).

For each C_{PoI} belonging to the suggested path, users can benefit from the availability of textual contents, photos, and other multimedia objects describing in detail the area and the interesting places related to their profile. Furthermore, the system allows a temporal browsing of multimedia collections related to the different ages.

Figure 17.2 shows some screenshots of the Android mobile application prototype. A user approaches the historical building *Palazzo Fruscione*: the app sends an appropriate alert, highlighting the point of interests on the map and providing the



The palace is named after its last owner and today is the Municipality. The building has several phases ranging from the eleventh-twelfth century to the present. A first group of flats was built in the Lombard curtis, next to the palace church, which in the eighth century was occupied by orchards and gardens. In Norman palace had only two floors, the third there was added between the end of the thirteenth century and the beginning of the fourteenth, while the fourth floor is a cant eighteenth century. The building is now used as a place of exhibition and welcomes exhibitions and events of various kinds.

Fig. 17.2 Screenshot of mobile app

user with a set of related services (ticketing, list of close restaurants, souvenir stores) or resources (descriptions, photos, social comments, and so on).

17.3 System Overview

17.3.1 Architecture

Figure 17.3 provides a functional overview of the proposed system in terms of its main components. As we can observe, PATCH presents the layered architecture typical of a pervasive system (see Chap. 2 [5]).

In the *data source layer*, each pervasive source is “wrapped” in order to extract the related information of interest that is then represented following a common data model. In particular, each *wrapper* is specialized for a particular kind of source (i.e., sensor networks, social networks, digital repositories, and Web data services) and must address all the interoperability issues, providing a set of functionalities to access data sources and gather all the desired data.

In the *data management layer*, data are stored in the *Knowledge Base* according to the data model and managed exploiting the *linked open data* (LOD) paradigm. In addition, semantics to be attached to data is given using proper annotation schemes, including ontologies, vocabularies, taxonomies, etc., related to the cultural heritage

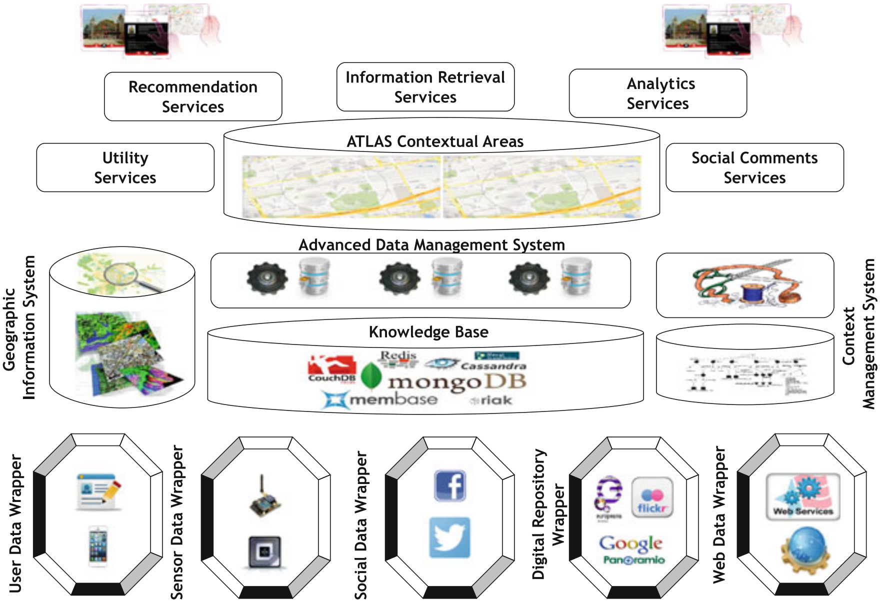


Fig. 17.3 PATCH architecture

domain. The Knowledge Base leverages different advanced data management technologies (e.g., Not only SQL (NoSQL) and relational systems) and provides a set of basic APIs to read/write data. The data management layer is further enriched with two additional components: a *geographic information system* (GIS) and a *context management system* (CMS). The GIS manages the entire atlas cartography offering a set of primitives to capture, store, manipulate, retrieve, analyze, and present all types of spatial or geographical data; the CMS performs data tailoring by exploiting context information and generates a set of georeferenced *contextual areas* for each user, possibly enabling a set of services on such data views.

As a basis for the *application layer*, our system provides a set of *services* that can be invoked by user applications. In its current version, PATCH offers:

- *Recommendation services*: both to suggest personalized cultural visiting paths to the users and to help them browse multimedia collections related to a given C_{PoI}
- *Information retrieval services*: to search information of interest using content-based facilities
- *Data analytics services*: to obtain useful statistics about a C_{PoI} or as support for data mining and pattern recognition applications
- *Social comment services*: to post comments and feedback about users' experience on social networks
- *Utility services*: for example, to inform users about weather conditions or how to book events as well as touristic attractions related to a given C_{PoI}

17.3.2 Functionalities

In the following, we describe in detail the main functionalities provided by the PATCH system.

17.3.2.1 Data Gathering from Pervasive Sources

As introduced, one of the most important functionalities offered by PATCH includes the capability of gathering different kinds of data from the various pervasive sources (see Chap. 2 [5]).

Concerning *sensor data*, we deal only with *wireless sensor network* (WSN) data sources. WSN capabilities can be exploited to monitor both the temperature and humidity of indoor or outdoor environments. In particular, we leverage the *PerLa* WSN management middleware, which abstracts a sensor network as a virtual distributed database system providing an SQL-like language for query formulation and data retrieval (as shown in Chap. 4 [15]). As an example, a PerLa query can be used (Algorithm 17.1) to retrieve the temperature at a given location, whenever there is a temperature change and the detected temperature exceeds 27 °C.

Algorithm 17.1 PerLa query for sensing if a location temperature exceeds 27 °C

```
CREATE OUTPUT STREAM SensedT (SENSID sid, LOCATION loc, FLOAT Tval, TIME
timest)
AS LOW:
EVERY 5m
SELECT sid, loc, Tval, timest
HAVING Tval ≥ 27
SAMPLING ON EVENT TemperatureChange
EXECUTE IF EXISTS (GPS, termometer)
```

Then, sensor data are periodically retrieved by the wrapper and converted into a serialized RDF format. For such kind of data, we have to deal with data stream management challenges (see Chap. 5 [12]); complex event processing techniques can be exploited to reduce the amount of information flow (see Chaps. 6 and 7 [6, 9]). An example of resource (in a *Turtle*-like format) that could be derived from such data sources is in the following.

Listing 17.1 Turtle resource derived from sensor data

```
<http://www.example.org/Measure/#32>
SensorAttributes:Type Temperature;
SensorAttributes:Time ‘12-Apr-14 14:20’;
SensorAttributes:Loc ‘40.76,-73.984’;
SensorAttributes:Value ‘34’;
SensorAttributes:Source WSNSensor:S01.
```

The example shows that in a given geographic area (identified by the GPS coordinates "40.76, −73.984"), a certain sensor (identified by the string "S01") has performed a temperature measure (whose value is "34 °") in a given temporal instant ("12-Apr-14 14:20").[3]

The meaning of all the metadata describing the measure (e.g., Type, Time, Location, Value, Source) can be retrieved in an annotation schema (*SensorAttributes*).

We note that user devices themselves, which can communicate the effective users' location by using the GPS technology or capture their preferences and needs, can also be considered as a different sort of source generating *user data* that have to be opportunely wrapped and coded in the described format.

As *social data*, the current prototype only considers information coming from Facebook. In particular, PATCH retrieves user comments and posts about a given C_{PoI} by exploiting the related *Graph APIs*. Some examples of resources that could be derived from such data source are described in the following.

[3] All the information about this measure is accessible via Web using the Uniform Resource Identifier (URI) http://www.example.org/Measure/#32.

Listing 17.2 Resource derived from social networks’ data

```
<www.facebook.com/...?comment\_id=1410540661>
SocialAttributes:Type Comment;
SocialAttributes:Time ‘14-Apr-14’;
SocialAttributes:Text ‘Wonderful’;
SocialAttributes:Topic The Tomb of the Diver;
SocialAttributes:User foaf:Vincenzo.Moscato.
```

This example shows that user Vincenzo.Moscato has expressed a comment (“Wonderful”) on a post about *The Tomb of the Diver*, the famous burial painting from the Greek period in Paestum on a specific date (“14 Apr 2014”).[4] The meaning of all the metadata describing the user activity (e.g., Type, Time, Text, Topic, User) can be retrieved in an annotation schema (*SocialAttributes*).

We collect *digital repositories’ data*, information describing cultural objects from online digital repositories (e.g., museums, libraries, theatrical foundations, digital archives, multimedia collections, etc.). Resources are often described based on the different annotation schemes and provided in XML- or RDF-based formats (as in *DBpedia*). The wrapper for this kind of sources can import such data descriptions and convert them into the described Turtle format according to the system data model as in the following.

Listing 17.3 Resource derived from digital repositories’ data

```
<http://www.example.org/CulturalObjects/Painting#1>
CulturalObjectAttributes:Type Burial Painting; \\
 CulturalObjectAttributes:Name ’The Tomb of the Diver’;
 CulturalObjectAttributes:Period Ancient GreekPeriod;
 CulturalObjectAttributes:Location Paestum;
 CulturalObjectAttributes:Description ‘The painted decoration ↩
     of the tomb called the diver, found in 1968, shows a ↩
     great moment of Greek painting, around 480 BC, ↩
     characterized by the same spirit of the painters of the ↩
     vascular severe style.’;
CulturalObjectAttributes:Subject.Concept ‘Real Life activity’.
```

The example shows a possible description of the famous painting *The Tomb of the Diver*, currently located in the Paestum Museum.[5] The meaning of all the metadata describing the painting (e.g., Type, Name, Author, Period, Description, Subject, etc.) can be retrieved in an annotation schema (*CulturalObjectAttributes*).

Multimedia Web resources (e.g., images) related to a given cultural object can be similarly collected using the wrapper facilities. In particular, the descriptions

[4] All the information about this activity is accessible via Web using the URI `www.facebook.com/...?comment_id=1410540661`.

[5] All the information about the cultural object is accessible via Web using the URI (http://www.example.org/CulturalObjects/Painting#1.

in terms of basic metadata are captured and stored within PATCH, while raw multimedia data can be opportunely linked and, in other cases, temporarily imported into the system for content-based analysis (see Chaps. 14 and 15 [1, 2]).

Other information of interest, such as that related to meteorological or traffic conditions, can also be captured using the available *Web data* services and represented in the described Turtle format.

Several examples regarding the *Salerno historical district* are described in the following.

Listing 17.4 Facebook resource referring to a painting in the Salerno historical district

```
<www.facebook.com/...?comment_id=1411540768$>
SocialAttributes:Type Comment;
SocialAttributes:Time '20-May-14';
SocialAttributes:Text 'Really amazing';
SocialAttributes:Topic Thermal Bath Painting;
SocialAttributes:User foaf:Francesco.Colace.
```

Listing 17.5 Multimedia resource describing a painting in the Salerno historical district

```
<http://www.example.org/CulturalObjects/Painting#14>
CulturalObjectAttributes:Type Public Painting;
 CulturalObjectAttributes:Name Thermal Bath Painting;
 CulturalObjectAttributes:Roman Imperial Period;
 CulturalObjectAttributes:Location Salerno;
 CulturalObjectAttributes:Description 'Fragment of wall fresco↩
      with candelabra motifs and red divisions. It is dated ↩
     between late I and early II AD and decorated a portion of↩
      the wall located in the central room of the Fruscione ↩
     Palace ground floor.';
CulturalObjectAttributes:Subject.Concept 'Decorative Floreal ↩
    Motif'.
```

In the previous examples, the user Francesco Colace left a comment ("Really amazing") on a post regarding the *Thermal Bath* painting (the description of which is reported in the second example) in *Palazzo Fruscione* on a given date ("20 May 2014"). This kind of decoration can also be found in the *San Pietro a Corte* area and is similar to some frescos discovered in the *Cappella di S. Anna* church. This is a simple composition of red panel alternating with stylized floral branches; it is not possible to identify other decorations between the squares, but there probably were some small figurative schemes. Other similar painting compositions are present in some buildings discovered under the *Tempio di Giove Toro* at Canosa (Julio-Claudian period) or in another chamber with mosaic floor discovered near *Archivio Storico del Banco di Napoli* in Via dei Tribunali, Naples (Augustan period with Flavian period modifications). Thus, our atlas using multimedia similarity facilities can suggest to visit other cultural places in Campania containing a similar heritage.

17.3.2.2 Data Management

The data gathered by the wrappers are stored and managed within the Knowledge Base of the system. Data are represented according to the *Europeana data model* (EDM), exploiting the LOD paradigm. In the EDM, each *cultural heritage object* (CHO)—the main entity of our model—related to a given C_{PoI} can be described at various levels of detail using different metadata schemes by means of the *aggregation* construct. As an example, social and sensor aggregations permit to associate/link to each CHO interesting environmental measures or users' comments from an online social network.

Two more EDM basic concepts, *place* (where the object is located) and *time span* (allowing to deal with different temporal views of the object), are considered in the model and related to the CHOs. Possibly, a certain number of Web resources (e.g., images, videos, texts, etc.) can be linked to each object. Finally, metadata semantics is provided by a set of annotation schemes (in XML, RDF, or Ontology Web Language (OWL) formats).

The data—represented as sequences of triples ($\langle$ subject, predicate, object $\rangle$) according to the described data model—are then stored in the Knowledge Base, based on several technologies:

- The data describing basic properties of CHOs (e.g., name, GPS coordinates, short description, etc.) are stored in a *key-value store*.
- The complete description in terms of all the metadata of a CHO together with user and sensor data is in turn memorized using a *wide-column store*.[6]
- The *document store* technology is used to deal with internal resources (multimedia data, XML and textual documents, etc.) that can be associated with an object.
- The relationships among cultural objects and users' social data are managed by means of a *graph database* (providing a collaborative and social network as shown in Chap. 8 [11]).
- The atlas cartography (each CHO is related to a given C_{PoI} and located in a given geographic area) is managed by the GIS, which provides functionalities to filter and visualize on a map the geographic area near a given C_{PoI} together with eventual suggested cultural visiting paths.
- The context description, in terms of *Context Dimension Tree*, is managed by the CMS, which allows the retrieval of useful data for users on the basis of the contexts that are tailored together with a set of services on the users themselves while browsing contextual areas in the atlas.
- Contextual data views can also be cached in *RDF stores* providing a SPARQL endpoint for the applications.
- The above applications, eventually, store the data into embedded *main memory databases*.

[6] We use a table for each kind of CHO having a column for each metadata; column values can be literals or URIs.

This heterogeneous Knowledge Base provides basic *Restful APIs* to read/write data and further functionalities for importing/exporting data in the most common diffused Web standards (e.g., JavaScript Object Notation (JSON)). The search of useful data for applications can be facilitated by using different *filters* that implement the right queries to the different databases.

We chose to adopt these groups of heterogeneous technologies in order to meet the specific requirements of applications dealing with huge amount of data. For example, social networking applications could benefit from graph database technologies because of their focus on data relationships. In turn, tourism applications could prefer more efficient technologies (key-value or wide-column stores) to quickly and easily access data of interest.

In addition, the data management techniques aiming to protect users' privacy (see Chaps. 3 and 10 [7, 8]) still have to be integrated into PATCH.

17.3.2.3 Context-Aware Data and Service Tailoring

The data retrieved from the Knowledge Base for the different users can be filtered using the approach described in Chap. 12 [4], which allows to define the information that is relevant for the current user needs and preferences (see also Chap. 11 [10]) depending on the context—represented by means of a Context Dimension Tree (CDT). This data is intentionally described by *contextual data views*, defined at design time and materialized on the user device when needed.

In a similar way, the system tailors, at run time, the subset of services really necessary in the current context, the activation of which is driven by a *context evolution Petri net* (CEPN), the formalism that models the evolution of a context instance into another one as a function of the conditions or services of the first context instance.

17.3.2.4 Multimedia and Recommendation Services

Following the guidelines in Chaps. 14 and 15 [1, 2], we can easily provide content-based information retrieval facilities for multimedia data. Furthermore, using recommendation techniques described in Chaps. 13 and 16 [13, 14], we can realize recommendation services with an automatic suggestion of visiting paths based on user profile and location, together with multimedia browsing capabilities.

17.3.2.5 Data Analytics

The big data analytics techniques summarized in Chap. 2 [5] are used to provide useful statistics for a given C_{PoI} (e.g., number of visiting users, average temperature and humidity for different periods of a year, etc.) or as a support to anomaly detection or for pattern recognition aims. Sentiment analysis techniques (see Chap. 9

[3]) can be further exploited to endow each C_{PoI} with a general user mood coming from the related comments in online social networks.

In addition, data mining techniques can be profitably exploited to discover significant relationships among users' profiles, general moods, and visiting paths. Such information could then be employed to enrich and improve user experience during C_{PoI}s' explorations.

17.4 Implementation Details

We tested the PATCH prototype for the two described case studies of Paestum and Salerno historical district. In the following, we report some implementation details concerning the customization of the developed system for the two areas.

Our data collection consists of a database of about 100,000 resources—mainly metadata descriptions, images, texts of CHOs coming from several digital repositories (i.e., Europeana, Flickr, Panoramio, Wikipedia)—related to all the main attractions of Paestum. For the Salerno historical district, we use scientific papers produced by historians from the University of Salerno and local materials (photos, videos) from the City Tourists Office. In addition, user comments and posts are gathered from Facebook by means of its *Graph API*. In addition, using the devices' GPS facilities and the *Google Weather* APIs, the system captures user location and some environmental parameters for a given area (number of persons and the related weather conditions).

Data are stored in the Knowledge Base, concretely implemented by means of the specific technologies belonging to the ones described in Sect. 17.3.2.2:

- We use *Voldemort* as key-value store containing basic information of all the managed CHOs, such as name, GPS coordinates, and short descriptions.
- The entire description in terms of metadata of a CHO is managed by the *HBASE* wide-column store together with user and sensor data related to the different C_{PoI}s.
- XML and multimedia documents related to CHOs are managed by the *MongoDB* document store.
- Multimedia data management has been realized using the *Windsurf* library.[7]
- Relationships among cultural objects and users' social data are maintained in the *AllegroGraph* database.
- User preferences (managed by *MongoDB* and *AllegroGraph*) are captured in an explicit manner by means of proper questionnaires or using information from social networks and in an implicit way considering users' session logs.

[7] http://www-db.deis.unibo.it/Windsurf/

- We exploit *Sesame*[8] as RDF store to memorize data views related to users' contextual areas.
- C_{PoI} spatial information has been managed by the *PostegreSQL* Object Relational DataBase Management System (ORDBMS) and its spatial extension *PostGIS*, leveraging the *Google Maps*.
- CDT and CEPN are managed using proper *Java* tools.[9]
- Basic analytics facilities are provided by *Hive* and *Impala* tools of the *Cloudera*[10] suite.
- Semantics of data can be specified by linking values of high-level metadata to some available ontological schema managed by *Sesame*.

All the functionalities provided by the Knowledge Base and services offered by the application layer have been implemented in *Java* and related technologies. Concerning the computational environment, we used the *SCOPE* distributed computing infrastructure at the University of Naples.[11]

Finally, for the moment a user can interact with the atlas by using an *Android* app. The presentation logic is based on apposite widgets. The client requests are elaborated by *JAVA Servlets*, and the results are sent to the client in the form of XML data.

17.5 Conclusions and Future Work

In this chapter, we showed an application and some examples bringing together all the data management pervasive technologies presented in this book.

In particular, we described the PATCH system—a portable prototype of a multimedia and social atlas of cultural points of interest, whose browsing is driven by the context. The system is characterized by several features that are typical of modern pervasive systems described in this book: (1) information gathering from distributed and heterogeneous pervasive data sources (sensor networks, social networks, digital libraries and archives, multimedia collections, Web data services, etc.); (2) context awareness, provisioning useful and personalized data and services, on the basis of users' preferences and of the surrounding environment; (3) advanced data management techniques and technologies for dealing with information variety,

[8] http://www.rdf4j.org

[9] Concerning the context management, we use two different kinds of CDTs: a *meta-CDT* representing a set of C_{PoI}s in a given geographic area with the related services and a *cultural CDT* with more specific information and services for each C_{PoI}.

[10] http://www.cloudera.com

[11] SCOPE consists of over 300 computing nodes (quad-core Intel Xeon 2.33 GHz processors with 8 GB RAM) communicating by dedicated optical-fiber channels.

velocity, and volume; and (4) advanced smart services such as retrieval, recommendation, analytics, and other utilities.

In addition, we described several real case studies implemented within DATABENC projects, related to an archaeological site (*Paestum Ruins*) and an urban area (*Salerno historical district).*

References

1. Amato, F., Greco, L., Persia, F., Poccia, S.R., De Santo, A.: Content-based multimedia retrieval. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
2. Bartolini, I., Patella, M.: Multimedia queries in digital libraries. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
3. Chang, S.K., Greco, L., De Santo, A.: Sentiment detection in social networks using semantic analysis: a tool for sentiment analysis and its application in cultural heritage realm. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
4. Colace, F., Moscato, V., Quintarelli, E., Rabosio, E., Tanca, L.: Context awareness in pervasive information management. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
5. Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L.: Pervasive systems architecture and the main related technologies In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
6. Cugola, G., Margara, A.: The complex event processing paradigm. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
7. De Capitani Vimercati, S., Foresti, S., Livagra, G., Paraboschi, S., Samarati, P.: Privacy in pervasive systems: social and legal aspects and technical solutions. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
8. Deliri, S., Albanese, M.: Security and privacy issues in social networks. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
9. Dell'Aglio, D., Balduini, M., Della Valle, E.: Applying semantic interoperability principles to data stream management. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
10. Koutrinka, G.: Data personalization. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
11. Moscato, V., Picariello, A., Subrahmanian, V.: Multimedia social networks for cultural heritage applications: the GIVAS project. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
12. Panigati, E., Schreiber, F.A., Zaniolo, C.: Data streams and data stream management systems and languages. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)

13. Pensa, R.G., Penta, A., Sapino, M.L.: Multimedia recommendation and delivery strategies. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
14. Sarwat, M., Bao, J., Chow, C.Y., Levandoski, J., Magdy, A., Mokbel, M.F.: Context awareness in mobile systems. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
15. Schreiber, F.A., Roveri, M.: Sensors and wireless sensor networks as data sources: models and languages. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)
16. Tommasetti, A., Vesci, M., Troisi, O.: The internet of things and value co-creation in a service dominant logic perspective. In: Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F.A., Tanca, L. (eds.) Data Management in Pervasive Systems. Springer, Berlin (2015)

Index

F. Colace et al. (eds.), *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, DOI 10.1007/978-3-319-20062-0