Advances in Data Management in the Big Data Era

Antonia Azzini[1], Sylvio Barbon Jr.[7], Valerio Bellandi[2], Tiziana Catarci[5], Paolo Ceravolo[2], Philippe Cudré-Mauroux[9], Samira Maghool[2], Jaroslav Pokorny[8], Monica Scannapieco[6], Florence Sedes[4], Gabriel Marques Tavares[2], and Robert Wrembel[3(✉)]

[1] Consortium for the Technology Transfer, C2T, Milan, Italy
antonia.azzini@consorzioc2t.it
[2] Università Degli Studi di Milano, Milan, Italy
{valerio.bellandi,paolo.ceravolo,samira.maghool,gabriel.tavares}@unimi.it
[3] Poznan University of Technology, Poznan, Poland
robert.wrembel@cs.put.poznan.pl
[4] IRIT, University Toulouse 3 Paul Sabatier, Toulouse, France
sedes@irit.fr
[5] SAPIENZA Università di Roma, Rome, Italy
catarci@diag.uniroma1.it
[6] Istituto Nazionale di Statistica (Istat), Rome, Italy
scannapi@istat.it
[7] Londrina State University (UEL), Londrina, Brazil
barbon@uel.br
[8] Charles University, Prague, Czech Republic
pokorny@ksi.mff.cuni.cz
[9] eXascale Infolab, University of Fribourg, Fribourg, Switzerland
pcm@csail.mit.edu, pcm@unifr.ch

Abstract. Highly-heterogeneous and fast-arriving large amounts of data, otherwise said *Big Data*, induced the development of novel *Data Management* technologies. In this paper, the members of the IFIP Working Group 2.6 share their expertise in some of these technologies, focusing on: recent advancements in data integration, metadata management, data quality, graph management, as well as data stream and fog computing are discussed.

Keywords: Data integration · Metadata · Data quality · Knowledge graphs · Data streams · Fog computing

1 Introduction

Data proliferation has been a reality for years and it is now considered the norm. Data of *multiple structures* and *large volumes* are produced at a *substantial speed*, challenging our ability to appropriately maintain and consume it. These three

characteristics are commonly known as the 3Vs (*Variety*, *Volume*, and *Velocity*) and describe the essential features of the so-called Big Data ecosystem [51]. The data types being produced and processed range from numbers, timestamps, short and large texts to time series, images, graphs, sounds, and videos, i.e., from fully structured to unstructured. The complexity of Big Data induced intensive research towards the development of new (or revisited) data models, data processing paradigms, and data processing architectures.

This complexity, in conjunction with the lack of standards for representing their components, computations, and processes, has made the design of data-intensive applications a failure-prone and resource-intensive activity. One of the reasons behind it can be identified in a lack of sound modeling practices. Indeed, multiple components and procedures must be coordinated to ensure a high level of data quality and accessibility for the application layers, e.g. data analytics and reporting. We believe that a major challenge of Big Data research requires - even more than developing new analytics - devising innovative data management techniques capable to deliver functional and non-functional properties like, among others: data quality, data integration, metadata discovery, reconciliation and augmentation, explainable analytics, data flow compliance or optimization.

Data Management research can address such challenges according to the FAIR principles. The goal is generating Findable, Accessible, Interoperable, and Reusable data. Methods, principles, and perspectives developed by the Data Management and Data Semantics community can significantly contribute to this goal. Solutions for integrating and querying schema-less data, for example, have received much attention. Standards for metadata management have been proposed to improve data integration among silos and to make data more discoverable and accessible through heterogeneous infrastructures. A further level of application of Data Management principles into Big Data technologies involves consistently distributing data processing across networks of interrelated data sources (sensors and data-flows), ensuring data quality and effective inference. The strong relationship between data quality and analytics also takes on research aimed at integrating Knowledge Graphs with advanced analytics powered by Machine Learning and Artificial Intelligence. Despite intensive research on data management techniques for Big Data (see [116] for more detail), unsolved issues and challenges persist. It motivated the members of the IFIP Working Group 2.6 (WG2.6): Databases[1] to further investigate the challenges and to publish a manifesto paper on Big Data Semantics [20]. The *Manifesto* revealed the limits of current technologies and identified concrete open problems. The WG2.6 *Manifesto* is not the only activity of the Working Group. WG2.6 ensues its tradition of promoting novel research areas in Data Semantics by means of research papers [2,22,48,67,93,94] and by organizing international research events. Since 2011, WG2.6 runs annually its main research event, i.e., International Symposium on Data-driven Process Discovery and Analysis (SIMPDA)[2]. In 2018 the

[1] https://www.ifip.org/bulletin/bulltcs/memtc02.htm.
[2] https://dblp.org/db/conf/simpda/index.html.

group initiated an international workshop on Semantics in Big Data Management (SemBDM).

Following the aforementioned WG2.6 *Manifesto*, in this summary paper, we overview current advances on the most significant topics presented in [116]), in the expertise area of the members of the WG 2.6. The topics include: leveraging knowledge graphs in the data integration process - Sect. 2, metadata discovery and management - Sect. 3, data quality - Sect. 4, data integration architectures - Sect. 5, graphs embedding - Sect. 6, processing and analyses of data stream in fog computing - Sect. 7, functional integration of relational and NoSQL databases - Sect. 8. Final remarks are reported in Sect. 9.

2 Knowledge Graphs for Data Integration

Knowledge Graphs have become one of the key instruments to integrate heterogeneous data. They provide declarative and extensible mechanisms to relate arbitrary concepts and data through flexible graphs that can be leveraged by downstream processes such as entity search [80] or ontology-based access to distributed information [27].

Integrating enterprise data into a given Knowledge Graph used to be a highly complex, manual and time-consuming task. Yet, several recent efforts streamlined this process to make it more amenable to large companies by introducing scalable and efficient pipelines [53,71,88]. The *XI Pipeline* [24] is a recent proposal in that context, which provides an end-to-end solution to semi-automatically map existing content onto a Knowledge Graph. We briefly discuss this pipeline below in Sect. 2.1—as an example of a state-of-the-art process to integrate data leveraging knowledge graphs—before delving into some of its applications in Sect. 2.2.

2.1 The XI Pipeline

An overview the XI Pipeline used to integrate heterogeneous contents leveraging a Knowledge Graph is given in Fig. 1. This pipeline focuses on semi-automatically integrating unstructured or semi-structured documents, as they are often considered the most challenging types of data to integrate, and as end-to-end techniques to integrate strictly structured data abound [75,87]. The Knowledge Graph underpinning the integration process should be given a priori, and can be built by crowdsourcing, by sampling from existing graphs, or through a manual process. The integration process starts with semi-structured or unstructured data given as input (left-hand side of Fig. 1) and goes through a series of steps, succinctly described below, to integrate the content by creating a set of new nodes and edges in the Knowledge Graph as output (right-hand side of Fig. 1).

Named Entity Recognition (NER) is the first step in the pipeline. NER is commonly used to integrate semi-structured or unstructured content, and tries to identify all *mentions* of entities of interest (e.g., locations, objects, persons or

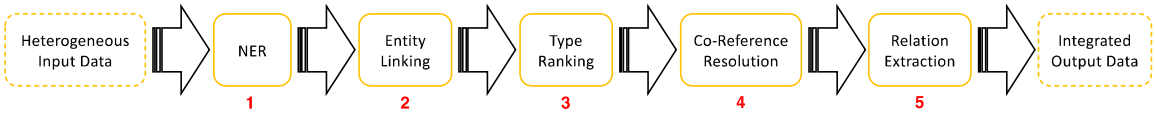

Fig. 1. The XI pipeline goes through a series of five steps to integrate semi-structured or unstructured content leveraging a knowledge graph

concepts) from the input content. This is typically achieved through Information Retrieval techniques using inverted indices over the Knowledge Graph to identify all relevant entities from the input content by leveraging ad-hoc object retrieval [105] as well as Big Data and statistical techniques [81].

Entity Linking naturally follows NER by linking the entities identified in the input to their correct counterpart in the Knowledge Graph. Various matching algorithms can be used in that sense, which can be complemented by crowdsourcing and human-in-the-loop approaches [29] for best results.

Type Ranking assumes that each entity in the Knowledge Graph is associated with a series of *types*. However, the types associated to a given entity in the graph are typically not all relevant to the *mention* of that entity as found in the input data. The XI Pipeline introduces the task of ranking entity types given their mentions in the input data [102] by leveraging features from both the underlying type hierarchy as well as from the (textual) context surrounding the mention [103]. The result of this process is a ranking of fine-grained types associated to each entity mention, which is invaluable when tackling downstream steps such as Co-Reference Resolution or Relation Extraction (see below).

Co-Reference Resolution identifies noun phrases (e.g., "the Swiss champion" or "the former president") from the input content that cannot be resolved by simple Entity Linking techniques. Such phrases are then automatically disambiguated and mapped onto entities in the Knowledge Graph by leveraging type information and the their context [82].

Relation Extraction finally attempts to identify *relationships* between pairs of entities appearing in the input content. The XI Pipeline resorts to Distant Supervision [91] in that context, leveraging a new neural architecture (the Aggregated Piecewise Convolutional Neural Network [90]) to solve this task effectively.

The outcome of the process described above is a set of nodes and links connecting mentions from the input data to entities and relations in the Knowledge Graph. As a result, the Knowledge Graph can then be used as a central gateway (i.e., as a *mediation layer*) to retrieve all heterogeneous pieces of data related to a given entity, type, relation or query.

2.2 Applications

Knowledge Graph integration can be used to solve many integration tasks in practice. The XI Pipeline, for instance, was successfully deployed in three very different scenarios:

– it was used to integrate research articles into a Knowledge Graph [1], in order to power pub/sub notifications related to specific research concepts, as well as research papers recommendations;
– it was also used to integrate and query series of heterogeneous tweets [104], which are otherwise very difficult to handle given their very short and noisy nature;
– finally, a particular instance of this pipeline was used in a large enterprise setting in order to integrate large-scale log data [65] and power a variety of applications ranging from job auditing and compliance to automated service level objectives, or extraction of recurring tasks and global job ranking.

3 Metadata Management: The Potential of Context

Metadata are simply defined as data about data [50] or information about information[3]. They are provided to help in the interpretation or exploitation of the data of interest, describing, locating, and enabling to retrieve them efficiently. Generally elicited from the contents themselves, they can be obtained from the context to enrich the value of a dataset. Indeed, metadata makes data sets understandable by both humans and machines, enabling interoperability, between systems with different hardware and software platforms, data structures, and interfaces, with minimal loss of data and semantics [25]. Fostering searchability metadata facilitates the integration of legacy resource and organizational silos or isolated applications. Metadata can help to comply with security and privacy requirements. Indeed, the core data remaining safe and protected, only metadata are transmitted across the network. Searchable data also helps to minimize data transfer paving the way to *sustainability*.

3.1 From Multimedia Contents to Metadata Management: The Example of Social Interaction Analysis

Multimedia contents have to be acquired and stored in real-time and in different locations. In order to efficiently retrieve the desired information, centralized metadata abstract, i.e., a concise version of the whole metadata, that locates some multimedia contents on remote servers, can be computed. The originality of this abstract is to be automatically built based on the extracted metadata. In [57] we presented a method to implement this approach in an industrial context and illustrated our framework with current Semantic Web technologies, such as RDF and SPARQL for representing and querying semantic metadata. Some experimental results, provided in order to show the benefits of indexing and retrieving multimedia contents without centralizing multimedia contents or their associated metadata, proved the efficiency of such a metadata abstract.

The metadata extraction is the most resource-consuming process in the management of multimedia collections. This raises the problem of the efficient

[3] https://csrc.nist.gov/glossary/term/metadata.

management of these large data volumes while minimizing resource consumption. User's constant interactions with multimedia contents and metadata complicate this management process. Issues about metadata management have been fixed by integrating "extra" information enrichment at different levels, each one relying on a layer. The metadata model matches the most widely used metadata standards, flexible and extensible in structure and vocabulary. In a multimedia management system, the indexing process is the most resource-consuming, through algorithms that extract metadata, whereas, in conventional systems, indexing implements a fixed set of indexing algorithms, without considering the resource consumption and user's changing needs. The user's needs are specified in his queries. In order to limit the metadata volume and on the other to reduce the resource consumption, we propose to split the indexing process into two phases: first time, at the contents acquisition time (i.e., implicit indexation), and, a second time, if necessary, at the query execution time (i.e., explicit indexation), the indexing algorithms being dynamically determined according to the required metadata.

Figure 2 shows an architecture for a framework based on a *holistic approach* that integrates multimodal heterogeneous cues and contextual information (complementary "exogenous" data) in a dynamic and optional way according to their availability or not. Such an approach allows the analysis of multi "signals" in parallel (where humans are able only to focus on one). This analysis can be further enriched from data related to the context of the scene (location, date, type of music, event description, etc.) or related to individuals (name, age, gender, data extracted from their social networks, etc.). The contextual information enriches the modeling of extracted metadata and gives them a more "semantic" dimension. Managing this heterogeneity is an essential step for implementing a holistic approach.

The automation of social interaction capturing and observation using non-intrusive devices without predefined scenarios introduces various issues we mentioned above namely (i) privacy and security, (ii) heterogeneity, and (iii) volume.

The proposed approach manages heterogeneous cues coming from different modalities as multi-layer sources (visual signals, voice signals, contextual information) at different time scales and different combinations between layers. The approach has been designed to operate without the need for intrusive devices, in order to ensure the capture of real behaviors and achieve the naturalistic observation. We have deployed the project on OVALIE platform which aims to study eating behaviors in different real-life contexts. To handle the high variety of the social cues, we propose a comprehensive (meta)data model for the visual nonverbal cues [83]. This model consists of four groups of entities: (i) acquisition group to store the used sensors' metadata (e.g., owner details, model number, transmission mode, data format, etc.); (ii) experiment group used to store the experiment's description including title, data, responsible person, and location, also the list of algorithms that are used to extract the social cues; (iii) video group used to store metadata related to the recorded video such as segments start/end timestamps, and frames information; and (iv) features group

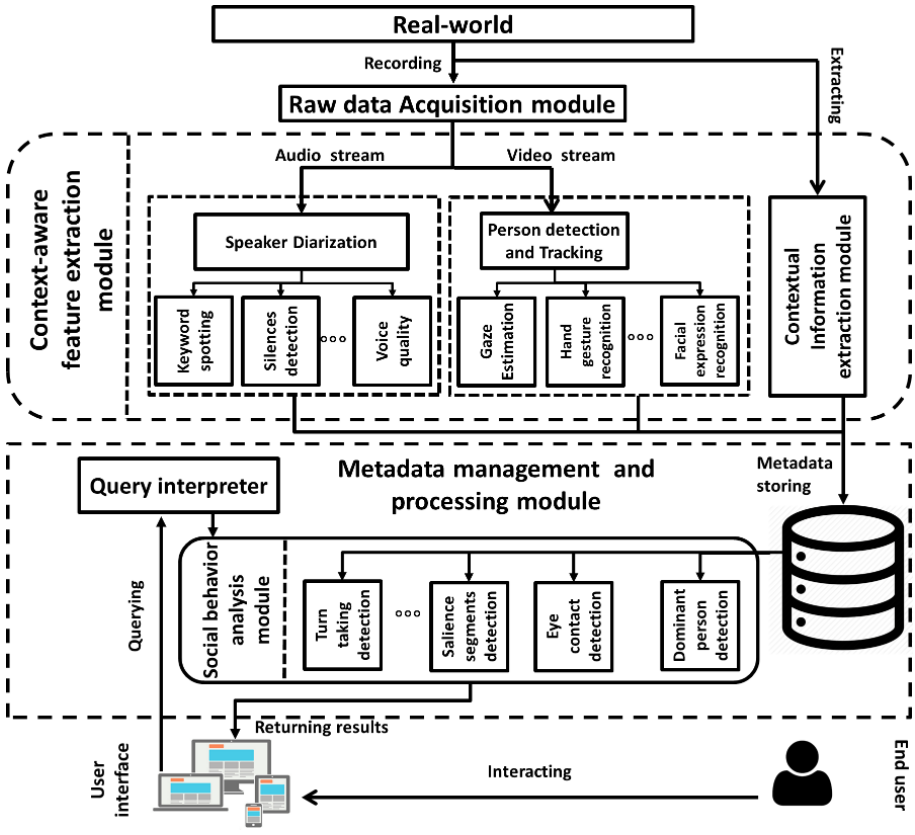

Fig. 2. Social interaction analysis framework architecture

to store the extracted social cues for each detected person in a given conceptual frame (conceptual frame is multiple frames that have a common timestamp and have to be analyzed together) as shown in the second figure. This generic data model shows the relationships between experiment, acquisition, video, and feature groups of entities, which are color-coded as green, orange, yellow, and gray in Fig. 3.

3.2 Internet of Things and Metadata for Trust Metrics

Internet of Things (IoT) is characterized by a high heterogeneity at different levels, and we mentioned how metadata models match this heterogeneity : (i) from the device level, a set of heterogeneous devices with dissimilar capabilities from computational and communication standpoints. Identifying, addressing, naming, and managing such devices in a standardized way is the first challenge. (ii) from a network-centric perspective, communication and interaction through various networks using different communication protocols (iii) from a data-centric vision, IoT is about exchanging massive amounts of data. It is essential to provide data with standardized formats, models and semantic descriptions, to support automated reasoning. As we said, optimization of energy and network bandwidth usage becomes an issue. As a matter of sustainability, metadata models allow for extensions and different types of metadata in different domains: the meta-

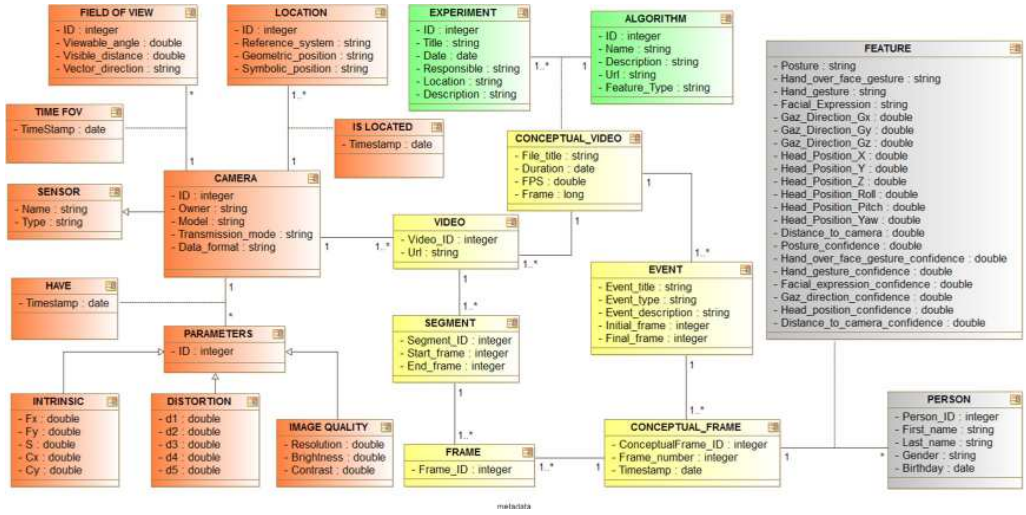

Fig. 3. Metadata meta modeling

data model should be flexible enough to cope with the evolutionary applications, devices, and needs, with energy-aware requirements.

4 Data Quality in the Post-truth Era

When people think about *data quality*, they often reduce it just to *accuracy*, e.g., the city name "Chicago" misspelled as "Chcago". However, data quality is more than simply accuracy: other significant dimensions such as completeness, consistency, and currency are necessary in order to fully characterize it. [9] provides a deep overview of which dimensions define data quality and describes several historical approaches to data quality characterization such as [112].

4.1 From Data Quality to Information Quality: An Increased Complexity of Quality Characterization

Most of the efforts paid to define data quality are related to structured data. However, a vast amount of realities is instead represented by types of information that are not structured data: a photo of a landscape, a map and a descriptive text in a travel guide, newspaper articles, satellite imagery etc. Dimensions for structured data are closely related to inner characteristic and properties of the underlying data model. An example is given by the different types of completeness, defined with and without the open world assumption for the different types of structures of the relational model, namely the tuple, the column, the table, and the set of tables [85].

When dealing with data types beyond structured data, it becomes necessary to define new dimensions for data quality, and, it becomes then more suitable to talk about *information* quality rather than *data* quality. With information quality, quality characterization starts to be dependent on the specific data type, e.g. quality dimensions characterizing images, such as sharpness and noise, are

very different from quality dimensions characterizing maps, such as topological consistency and positional accuracy.

In addition to the dependency on data types, information quality can have some relevant domain specialization. For instance, textual data describing laws can be characterized by dimensions such as conciseness and unambiguity, while textual data of novels can be characterized by cohesion and coherence [9].

4.2 Information Quality in Modern Society

We are living in a datafied society [66], where there is a relevant paradigm shift: from "primarily designed" data that were modeled and stored within information systems with a defined semantics, to "secondary produced" data, i.e. data resulting from interactions with devices or passively produced by systems, like sensors data and Web logs. These data are collectively referred to as Big Data: in addition to the 3Vs characterizing Big Data, a fourth V is particularly important i.e. *Veracity*. Veracity directly refers to information quality problems: with the huge volume of generated data, the fast velocity of arriving data, and the large variety of heterogeneous data, the quality of Big Data is far from being perfect [18]. A notable example of the need for assessing Veracity is the fake news phenomenon, causing the spreading of misinformation across social media users [44]. A further example is provided by information available on the Web, for which the issue of assessing veracity is particularly important. In [62], for instance, several sources providing the same information in two domains on the Web, namely Stocks and Flights, are compared to discover inconsistencies and inaccuracies and hence point to the "true" data. As another example, in [34], an approach for assessing the veracity of linked data is presented with a proposal of a fact-checking framework for data modeled as RDF triples.

Quality of today information also includes a new relevant dimension, namely *data fairness*. Fairness can be defined in terms of lack of discrimination, i.e. treating someone differently. It is possible to distinguish between two categories of fairness: individual fairness, for which similar predictions are given to similar individuals and group fairness (also known as statistical fairness), for which different groups are treated equally independently of a particular race, gender, or sexual orientation [31]. A very famous example of group unfairness is the COMPAS algorithm used by the Department of Corrections in Wisconsin, New York and Florida that has led to harsher sentencing toward African Americans [7]. In order to achieve fairness, it is important to deal with bias in data, algorithmic and user evaluation. In [68], several examples of different types of bias impacting on fairness are provided, including social bias, e.g. if a review is influenced by different scores provided by other reviewers and there is hence a social influence experienced, or cause-effect bias, which can happen as a result of the fallacy that correlation implies causation. Notably, there are several cases of population bias for instance in the health domain, where the fallacy of predictions can be particularly serious. A notable field for data fairness is in relation to learning algorithms (data analytics and Machine Learning - ML) that base their predictions on training data and improve them with the growth of such data. In

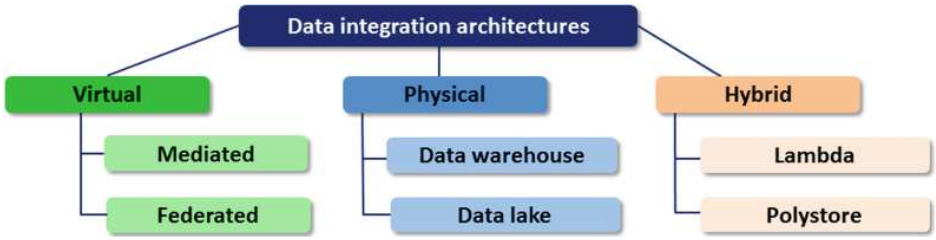

Fig. 4. The taxonomy of data integration architectures

a typical project, the creation and curation of training data sets is largely a human-based activity and involves several people: domain experts, data scientists, machine learning experts, etc. In other words, data-related human design decisions affect learning outcomes throughout the entire process pipeline, even if at a certain point these decisions seem to disappear in the black-box "magic" approach of ML algorithms. On the other hand, it is now gaining attention the fact that humans typically suffer from conscious and unconscious biases, and current historical data used in training set very often incorporate such biases, so perpetuating and amplifying existing inequalities and unfair choices. It is a still open problem to figure out concrete solutions on how to discover and eliminate unintended unfair biases from the training data sets and/or to create "by design" data sets that are natively "fair".

5 Data Integration Architectures for Standard and Big Data

Large companies typically store their data in heterogeneous storage systems, ranging from files to fully functional databases, further called *data storage systems* (DSSs). Querying such DSSs in an integrated way is challenging as the systems typically support different ways of querying, use different data models and schemas (even if designed in the same data model, e.g., relational). For more than 50 years researchers worldwide have dealt with this problem and have proposed a few data integration architectures. Their achievements are outlined in this section.

5.1 Data Integration Taxonomy

The taxonomy of Data Integration (DI) architectures is shown in Fig. 4. Three main categories are distinguished, namely: *virtual*, *materialized*, and *hybrid*. In the *virtual* architecture, data are stored in their original DSSs and are accessed on the fly, via an integration layer. In the *physical* architecture, data are ingested from DSSs and stored locally (materialized) in advance in an integration system. The *hybrid* architecture combines the functionality of the two aforementioned DI architectures, i.e., some data are integrated and accessed on the fly, whereas other data are pre-integrated and materialized.

5.2 Virtual Integration

Two types of *virtual* architectures have been proposed, i.e., *federated* [12,32] and *mediated* [115]. Their main feature is that data are integrated on the fly via an intermediate layer located between the user and DSSs. This layer is responsible for: (1) transforming source data models into the common one, typically the relational one, (2) decomposing user queries into sub-queries and routing them into appropriate DSSs, (3) transforming the sub-queries into executable snippets on each DSS, (4) transforming and integrating results of the sub-queries.

The main difference between the *federated* and *mediated* architecture is that the first one is used to integrate databases and it uses more components in the intermediate layer (e.g., a transforming processor, a component schema, a filtering processor, and an export schema). The *mediated* architecture is applied to integrating also other DSSs than databases. It uses two main components as the integration layer, i.e., a wrapper and a mediator.

5.3 Physical Integration

Two types of such architectures, accepted as de-facto industry standard, have been proposed, i.e., a *data warehouse* (DW) [49,106] and a *data lake* (DL).

In the DW architecture, the integration is implemented by means of the Extract-Transform-Load (ETL) layer where the so-called ETL processes (workflows) are run. They are responsible for: (1) ingesting data from data sources, (2) transforming heterogeneous data into a common data model and schema, (3) cleaning, normalizing, and eliminating data duplicates, (4) loading data into a central repository - a data warehouse.

The widespread of Big Data induced another architecture - a Data Lake, used to implement a staging area. The DL is a repository that stores a vast amount of heterogeneous data ingested in their original formats [84,100]. Then, the content of a DL processed by ETL processes to build cleaned, homogenized, and integrated data repositories. To this end, rich and well-organized metadata annotations are needed to provide a precise description of data [70]. Typical storage for DLs is based on Hadoop, Azure, or Amazon S3.

5.4 Hybrid Integration

For storing Big Data, alternative data stores have been developed, including key-value, column family, document, and graph stores, commonly referred to as NoSQL stores. Multiple systems produce data at much larger speed than before - we refer to such systems as streaming data sources (e.g., sensors, medical monitors) [52]. The variety of data formats and speed of data production by DSSs, makes the data integration process very challenging. To ease this process, two novel data integration architectures have been developed recently, i.e., the *polystore* and the *lambda* architecture.

The Polystore Architecture. Figure 5 shows a general architecture of a polystore. Heterogeneous DSSs (denoted as *DS1*, ..., *DS4*) are connected to the *integration middleware* via dedicated interfaces (drivers) (denoted as *Interface1*, ..., *Interface4*). They offer the functionality of wrappers (as in the mediated architecture). The *integration middleware* makes available all these DSs for querying via a common schema. This middleware offers functionality similar to a *mediator*. These features make the polystore a virtual DI architecture. Typical types of DSs integrated into the polystore architecture include: relational, array, key-value, graph, stream, DFS [96].

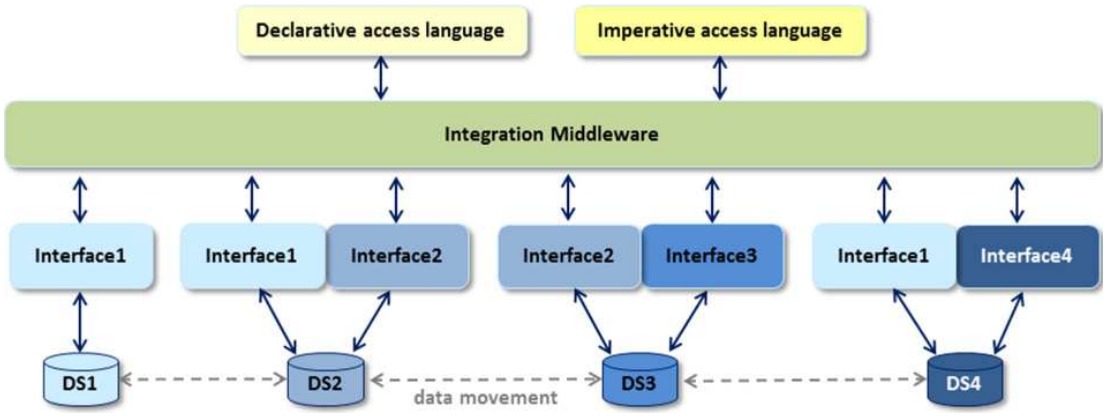

Fig. 5. An overview of the *polystore* architecture

Three features distinguish polystores from other architectures. First, the *integration middleware* allows to query DSs using different languages, typically a declarative one (SQL-like) and an imperative one. Second, data sets can be relocated from one DSS into another, to improve the performance of queries. Technically speaking, data sets can be either copied (replicated) or moved into a DSS where a given query will be executed faster. For example, let us assume that *DS1* is a relational database and *Interface1* exposes a relational data model and SQL; *DS2* is a key-value store and *Interface2* exposes a procedural interface for searching. Notice that *DS2* can also be accessed via *Interface1*. If an SQL query is executed on both *DS1* and *DS2* via *Interface1*, the global optimizer may decide to move or copy data from *DS2* into *DS1*. This feature makes the polystore a materialized DI architecture, therefore classified as a hybrid DI architecture. Third, a given DS may be queried via more than one interface and thus, provide data in different data models.

Examples of polystores include among others: BigDAWG [30], Polypheny-DB [109], CloudMdsQL [55], Estocada [5,14] (an overview of such systems is available in [96]).

The Lambda Architecture. The lambda architecture combines an architecture for collecting data in a batch mode (cf. the batch layer in Fig. 6) and collecting fast arriving data (cf. the real-time layer in Fig. 6) [92]. The purpose of

lambda is to be able to combine slowly arriving data with fast arriving data for analysis. The bath layer can be instantiated by a standard DW architecture, whereas the real-time layer can be instantiated by a standard stream processing architecture. Both layers are integrated using the serving layer, typically implemented by means of materialized and virtual views.

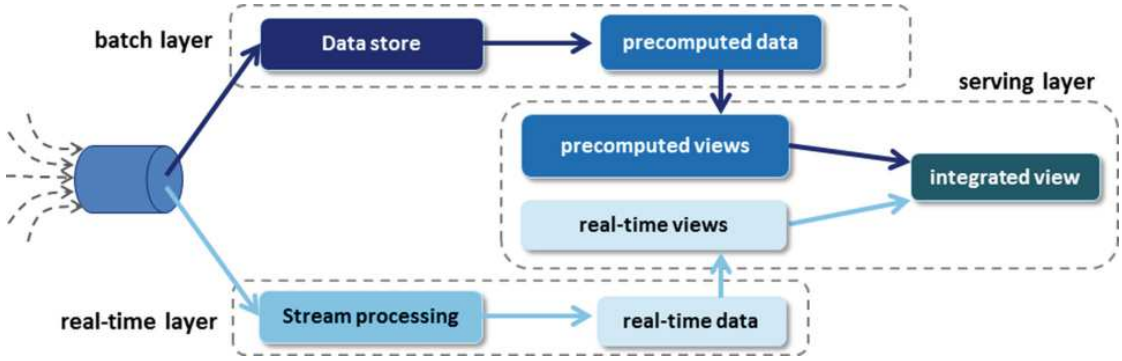

Fig. 6. An overview of the *Lambda* architecture

Typical components used in this architecture include: (1) in the batch layer - relational DBMSs or NoSQL DSSs, (2) in the real-time layer - Kafka, Kafka Streams, Spark, and Cassandra.

To sum up, the data integration architectures developed for Big Data extend the ones developed earlier for standard DSSs. Despite substantial research and technological advancements in this field, there still exist multiple unsolved problems. Some of the problems include (1) development of metadata standards for interoperable data DI architectures (in the spirit of [52]), (2) query optimization techniques in the highly heterogeneous and distributed architecture, composed of various data models, physical storage models, and hardware (in the spirit of [38]), (3) performance optimization of data integration processes (in the spirit of [3,89].

6 Graph Embeddings

Data Graphs, consisting in networks of nodes and relationships, are largely adopted in support of distributed and modular data modeling and data analytics procedures. Learning and predicting new links in social media [35], protein-protein interaction [101], distributed monitoring of business process [60] distributed ledgers [56] are among examples of applications leveraging on graph data models.

A limit of data graphs is that the information they carry cannot be readily applied in Machine Learning (ML) due to a mismatch at the representation level. While ML typically works with feature vectors, where a single instance is described by a flat set of features, data graphs link the features related to a data instance using a network of interconnected nodes. Graph embedding helps

in handling this mismatch by transforming nodes, edges, and their features into a vector space while partially preserving properties of their original graph structure. The general goal is that nodes connected in the graph are kept closer in the embedding space but task dependent transformation strategies are studied to meet the requirements of different ML algorithms. Three common steps are needed in defining an embedding procedure: (i) define an encoding procedure mapping the nodes of a graph into embeddings, (ii) define a decoding procedure to extract node properties from the embeddings (i.e. node neighborhood or labels) (iii) optimize the decoding results, typically by multiple iterations, to get from the embeddings results that are as much as possible equivalent to those of the original graph. Figure 7 schematizes this idea and represents the three most mentioned methods in the literature.

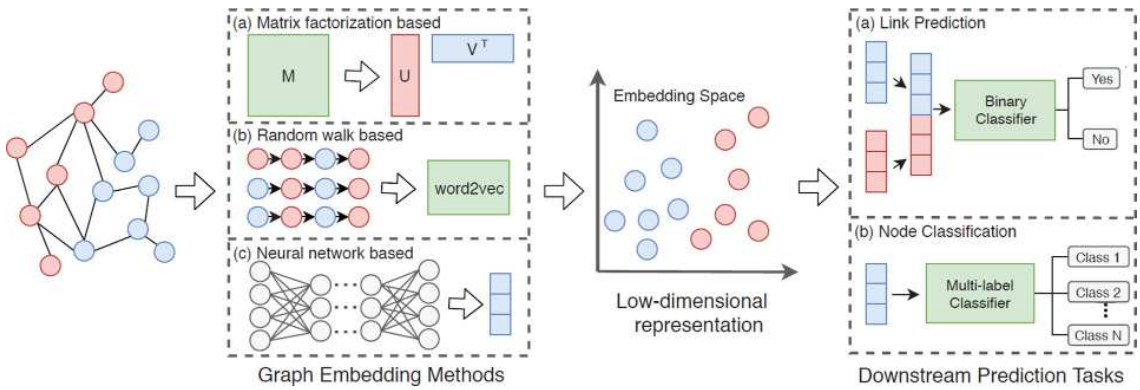

Fig. 7. A schematic comparison of three proposed algorithms in graph embedding adapted from [120] page 2

Matrix Factorization-Based Algorithms. These algorithms focus on factorizing the matrix representation of nodes relations in order to obtain the embedding. For some property matrices, the factorization problem is abstracted in eigenvalue decomposition (e.g. Laplacian matrix [6]) while for unstructured matrices the goal is solving an optimization problem (e.g. gradient descent methods). For example, SocDim [98] factorizes the modularity matrix and the normalized Laplacian matrix; NEU [119] factorizes similarity matrices that encode an higher order of the adjacency matrix, Laplacian Eigenmaps [10] adopt pairwise similarity and imposed a quadratic penalty function, MDS [47] adopt the Euclidean distance between two feature vectors as pairwise similarity. Other proposals [108, 114] aim to learn pairwise distances by semidefinite programming procedures. For preserving high-order proximity, HOPE [72] factorizes several matrices based on different similarity measures while GraRep [16] factorizes a matrix that is related to the k-step transition probability matrix. Multiple objective functions can be adopted in order to compensate the advantages/disadvantages of different techniques [4, 15].

Random Walk-Based Algorithm. The idea is to encode the coordinates such that the similarity of two nodes in the embedding space equals the probability

that the two nodes co-occur on a random walk over the network. Deepwalk [73] and Node2vec [42] are two popular random walk strategies while the former uses a deterministic approach and the latter a probabilistic one. Other techniques run random walks on modified versions of the original network. Walklets [74] modify the random walk strategy used in DeepWalk by skipping over some nodes in the graph. LINE [97], to embed larger information, optimizes a designed objective function based on 1-hop and 2-hop random walk probabilities that preserves both the local and global network structures.

Neural Network-Based Algorithms. This approach uses neural network to construct deep encoders, i.e. the output space is non-orthogonal to the input space. For example, SDNE [111] and DNGR [17] use deep autoencoder. Different approaches that adopt convolutional neural network differ in the way of formulating convolution-like operations on graphs [13,28,45,86]. GCNs [54] learns how to propagate information across the graph to compute node features and generate node embeddings based on local network neighborhoods. GraphSAGE [43] uses graph convolutional networks to generalized neighborhood aggregation. It is worth noting that all these deep encoders can be combined with similarity functions obtained by random walk methods.

Hybrid Methods. Sometimes multiple methods are combined. In [121] the authors leverage the heterogeneous information in a Knowledge Graph (KG) to improve recommendation performance using TransR [64] for network embedding, and autoencoders for textual and visual information. KR-EAR [63] constructs a relational triple encoder (TransE [113], TransR [11]) to embed the correlations between entities and relations, and an attributional triple encoder to embed the correlations between entities and attributes. TransE and TransH models build the embeddings by putting both entities and relations within the same semantic space while TransR build the embeddings putting entities and relations in separate spaces.

The Explainability Challenge. Graph embeddings represents a powerful tool for conciliating the representational power of data graphs with the powerful analytics of ML. This approach has been however criticized for creating an additional level of complexity that isolates the input and the output data by a black-box procedure that hamper the explainability of ML results. The community is then trying to get explainable ML methods. A promising approach is linking the layers of a neural network with symbols of a *lingua franca* [37]. In this context, Knowledge Graphs could play a crucial role in supporting explainability especially with embeddings methods that exploit the layer of the neural network for generating the vector space. In a neural network architecture where input features, hidden layers, computational units, and predicted output are mapped to the entities of a KGs, results could be easily translated in terms of these entities. Whilst most of the existing methods in KG embeddings such as TransE, TransH, and TransR, are not explainable by design, an increasing number of recent studies concentrating on transparent explainable models [8,33,95,117].

7 Data Stream Processing and Analytics in Fog Computing

The advent and development of the Internet of Things (IoT) encompasses that everything can connect to the Internet, generating data streams from the events recorded by sensors and probes. Data stream processing imposes specific constraints on data management [39]. If the conventional processing model clearly differentiates between data storage and data processing stages, handling data streams implies processing in real-time weakening the role of central data storage and data management procedures. The needs covered by *Data Management* are however still in place with data streams, or even emphasized. Heterogeneous and decentralized sources require *data integration*, *noise filtering*, *concept drift detection*, *uncertainty*, and *data quality management* [99]. That is to say, data management is not canceled, but it is simply decentralized as it has to follow data at the production and transmission levels. The *Fog Computing* paradigm represents, in this sense, an important advancement for the Cloud Computing solutions often coupled with data stream processing.

7.1 Recent Advances in Stream Processing Algorithms

Great work has been done to address the challenges of data stream processing by designing innovative algorithms and data analytics procedures. We present the most relevant contributions in the area by distinguishing three families of algorithms based on the main goal they address. In particular, we have highly accurate solutions, lightweight algorithms, and robust methods.

Highly Accurate Solutions. Considering the requirement of high predictive performance and the huge volume of data provided by IoT ecosystems, Deep Learning (DL) solutions arise as a suitable processing tool. DL architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short Term Memory (LSTM) can learn hidden features from the raw and noisy data [69]. Some recent frameworks support the vastest DL modeling, e.g., ADLStream [58] demonstrated superior predictive performance than traditional statistical temporal series approaches and most well-known stream mining algorithms.

Lightweight Algorithms. A restricted computational cost scenario composed of lightweight devices demands reducing memory costs. In this category, we group the algorithms capable of reaching a competitive predictive performance with low memory cost without compromising processing time. An example is the Strict Very Fast Decision Tree [23], an algorithm that minimizes tree growth, substantially reducing memory usage, leading to being used as a base learning into ensemble solutions.

Robust Methods. Concept drifts detection, novelty pattern recognition, and forget outdated concepts are requirements for building a robust data stream

method. In this category, concept drift detectors embedded into ensemble methods have been stood out. This sophisticated combination takes advantage of non-parametric drift detection methods based on Hoeffding's bounds [36], for triggering model updating. Ensembles are highly predictive and effective methods for mitigating concept drift, paving the way for hybrid architecture for supporting robust classification procedures with non-stationary data streams.

7.2 The Fog Computing Paradigm

If the IoT requires that objects can continuously communicate [61], to ensure timely data processing there is a need to continuously reconfigure the computational resources used for processing data streams [46]. Up to now, mostly cloud-based computational resources have been utilized for this. However, cloud data centers are usually located far away from IoT sources, which leads to an increase in latency since data needs to be sent from the data sources to the cloud and back. Today a new emerging architectural paradigm is changing this processing model. *Fog Computing* is a cloud technology in which terminal-generated data does not load directly into the cloud, but is instead pre-processed by decentralized mini data centers [26]. The concept involves a network structure extending from the network's outer perimeter in which data generated by IoT devices is sent to the public cloud central data endpoint or to a private data processing center (private cloud). With the advent of Fog Computing, it is possible to perform data processing and data management in the cloud as well as at the edge of the network, i.e., exploiting the computational resources offered by networked devices (Fig. 8). To better understand how to extend this idea can be applied we need to review the stages composing data stream processing pipelines. In fact, we can model a classical data streaming pipeline into three main stages [20].

Data Preparation. This stage relates the cleaning and transforming raw data prior to processing and analysis. It is an important step and often involves reformatting data, making corrections to data, and combining multiple source sets to enrich data. Data preparation is often a lengthy undertaking for data professionals or business users, but it is essential as a prerequisite to put data in context, turn it into insights, and eliminate bias resulting from poor data quality. For example, the data preparation process usually includes standardizing data formats, enriching source data, removing outliers, and/or transform the data format [21,107].

Data Analytics. This stage aim at extracting knowledge from the data acquired by the sensors. *Descriptive Analytics* allows representing the facts recorded. *Predictive Analytics* allows performing data analysis in order to draw predictions on future outcomes. *Prescriptive Analytics* combines data analysis with the ability to take and manage decision-making procedures. Prescriptive Analytics provides strategic indications or operational solutions based on both Descriptive Analysis and Predictive Analysis [19].

Visualization and Reporting. This stage aims at presenting the information carried by data, using visual elements like charts, graphs, and maps. Data visu-

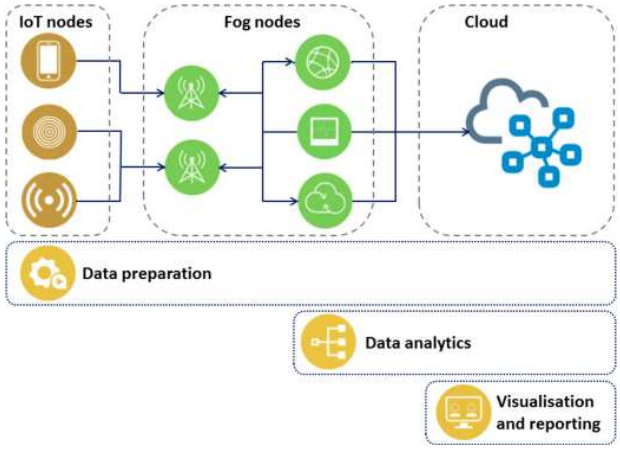

Fig. 8. Fog computing and its relationship with data processing stages

alization tools provide an accessible way to see and understand trends, outliers, and patterns in data [118]. In the world of Big Data and IoT, data visualization tools and technologies are a key factors to analyze massive amounts of information and to make data-driven decisions.

Data Stream Pipelines in the Fog. A key design decision to be taken in modeling data stream processing and analytics is how distributing these stages in the edge area (in the "Fog"). As a consequence, it becomes crucial to understand how data management affects the results of complex and distributed processing analytics. Further work is then required to understand the implications of unifying and interconnecting decentralized procedures with their possibly conflicting requirements and boosting effects. For example, several filtering or aggregations can be applied directly in the edge area. Figure 8 illustrates this concept by showing that some data processing stages can be applied directly to the IoT or Fog nodes. Data Preparation applies to the IoT, Fog, or Cloud layers. Data Analytics applies to the Fog and Cloud layers, while Data Visualization and Reporting apply to the Cloud layer only.

8 Integration of Relational and NoSQL Databases Functionally

Relational and NoSQL databases (DBs) contained in one integrated architecture require an infrastructure involving data and software of both transactional and analytical types (cf. Sect. 5). A particular case of integration of relational and NoSQL DBs concerns graph DBs and document DBs using JSON format. One tendency is to use multi-level modeling approaches involving both relational and NoSQL architectures enabling their simple integration [78].

Today, NoSQL DBs are considered in contrast to traditional RDBMSs products. On the other hand, yet other modeling approaches are possible, e.g., a functional approach. We can mention Gremlin - a functional graph query language that provides traversal operators/functions chained together to form path-like

expressions. Gremlin is supported by many GDBMSs. Significant works using a functional approach to data management are contained in [41].

The main aim of the section is to use a functional approach introduced in [76] to modeling both relations and data structures occurring in NoSQL DBs. For graphs, we will use a *(labelled) property multigraph model*. Both nodes and edges are defined by a unique identifier (Id). Properties are simply single-valued attributes. A property graph is represented by a set of typed partial functions. The functions considered will be of two kinds: single-valued and multi-valued. Regardless of the fact that the graphs are considered schema-less in NoSQL DBs, we will use graph DB schemas. The associated model of JSON data based on JSON Schema can deal with data also as functions. The relations in a relational DB can be considered also as typed functions. Some possibilities for integration on the level of a common query language are presented and discussed.

For graph querying in today's practice, we can consider GDBMS Neo4j and its popular query language Cypher. For relational DBMSs, of course, we assume SQL. Querying JSON documents can be done, e.g., with as in the most popular document store MongoDB. Then, a typed lambda calculus, i.e., the language of lambda terms (*LT language*), can be used as a data manipulation language. More integration details can be found in [79].

8.1 Functional Approach to Data Modeling and Querying

The functional approach used here is based on a typing system. For our purposes, we use elementary types and a number of structured types. Typed functions appropriate to modeling real data objects are attributes viewed as empirical typed functions that are described by an expression of a natural language [76]. To propose them means that we construct a conceptual or DB schema.

We assume the existence of some (*elementary*) *types* S_1,...,S_k ($k \geq 1$) constituting a *base* **B**. Always *Bool* ϵ **B**. It allows to model sets (resp. relationships) as unary (resp. n-ary) characteristic functions. Table 1 presents basic structured types for typing objects in relational, graph, and JSON data model, respectively.

Table 1. Types used in relational, graph, and document data models

Type name	Type structure	Relational model	Graph model	Document model
Functional	(S:$R_1, \ldots, R_n$)	Yes, n≥1, S=Bool	Yes, n=1	Yes, n=1
Tuple	($R_1, \ldots, R_m$)	No	Yes, S is a tuple, m≥1	No
Set	{$R_1, \ldots, R_m$}	No	No	Yes, n≥1
Array	[$R_1, \ldots, R_m$]	No	No	Yes, n≥1

Supposing, e.g., entity types *Movie* and *User*, then the expression "the movies rated by a user" denotes a ((*Bool:Movie*):*User*)-object, i.e. a (partial) function f:*User* → (*Bool:Movie*). At the schema level we write *Rates*/((*Bool:Movie*):*User*). GDBMSs can use attributes of types (R_1:R_2) and (($Bool$:R_1):R_2), respectively, where R_1 and R_2 are entity types. Properties

require to use tuple types, e.g., *Movie*/((*Title, Director*):*Movie*). A relational DB can contain *Actors*/(*Bool: Name, Title, Role*).

Logical connectives, quantifiers and predicates are also typed functions, e.g., **and**/(*Bool:Bool,Bool*), + is (*Number: Number, Number*)-object. The aggregation function COUNT_R is of type (*Number:(Bool:R)*). A manipulation language for functions – *LT language*, uses terms based on *applications*, *lambda abstractions*, *tuples*, and *components of tuples*. Document DBs use also terms like *set elements*, *arrays*, and *elements of array*. Typically, lambda abstractions serve as a tool for expressing queries, i.e., they provide answers – typically relations. The query "Give a set of couples associating to each film directed by Burton the number of actors who acted in it" can be expressed but the term

$\lambda\ t,\ p\ (p = \text{COUNT}(\lambda\ a\ \exists\ t,\ r\ \text{Actor}(a,t,r)$ **and** $\exists m\ \text{Movie}(m)(t,\text{'Burton'})\ ))$

A more advanced approach enables to construct more complicated data structures, i.e., new graphs or documents [77]. JSON types can be extended to regular expressions.

8.2 Integration of Relational, Graph, and Document Databases

For example, [40] offers three ways of integration of relational and NoSQL DBs: *native*, *hybrid*, and *reducing to one option, either relational or NoSQL*. In [68], possible approaches are categorized as a *polyglot persistence*, *multi-model approach*, and *schema and data conversion*. We use the multi-model approach, particularly *multi-level modelling* with a *special abstract model*. Obviously, there are more approaches, e.g., *NoSQL-on-RDBMS*, *SQL-on-Hadoop*, and *ontology integration* [79]. The approaches are mutually related and rather describe particular cases with given NoSQL/relational DBs than more exactly specified categories.

A multi-model approach reminds a more user-friendly solution of heterogeneous DB integration as it was known in the context of heterogeneous relational DBs in the past. Here, we suppose that different DB models are behind participating DBs. An alternative for data processing with relational and NoSQL data in one infrastructure uses common design methods based on a modification of the traditional 3-level ANSI/SPARC approach. This multi-level modeling is most relevant for our multi-model approach.

One sub-approach of the multi-model approach is a *hybrid approach*, e.g. [110]. In this case, the query is executed on more data sources, relational and some NoSQL, but the additional layer is used to enable data integration. For example, a generalizable SQL query interface for both relational and NoSQL systems called Unity is described in [59]. Unity is an integration and virtualization system as it allows SQL queries that span multiple sources and uses its internal query engine to perform joins across sources.

In our case, LT queries sent to the integrated system are translated into queries compatible with the RDBMS (e.g., SQL) and GDBMS (e.g., Cypher), or JSON document store (e.g. MongoDB), respectively.

9 Conclusion

A widespread of huge data volumes of arbitrary structures encouraged new data management techniques for data processing. In this summary paper, we overviewed current advances in some of these techniques, being in the expertise of the members of the IFIP Working Group 2.6: Databases. Despite traditional *Data Management* has been considered inappropriate to the data processing techniques used in the Big Data ecosystem the current advancements make clear important challenges emerging in this area need to embrace a Data Management perspective.

Data integration typically requires a mediation layer to integrate heterogeneous content. In this article, we saw how *Knowledge Graphs* can be used as such a layer to integrate Big Data, using pipelines to automatically identify and relate pieces of data from the input content and map them onto entities and relations in a Knowledge Graph. The resulting integrated Knowledge Graph can then be used as a global abstraction to efficiently and effectively search, query or manipulate all heterogeneous sources centrally.

Initially designed for summarizing, describing, enriching (un)structured (big) data collections, *Metadata* help from now on to comply with larger requirements to adapt to new environments such as IoT, social interactions analysis, cybersecurity, and other. Indeed, the data itself remaining safe and protected, only metadata are transmitted and exchanged. Minimizing data transfer, exploiting, and improving data locality is a crucial concern, paving the way to *sustainability*: metadata models allow for flexibility and context-awareness in any domain and at various levels, with privacy, security and energy-aware requirements.

Quality of today information also includes a new relevant dimension, namely *Data Fairness*. Fairness can be defined in terms of lack of discrimination, i.e. treating someone differently. In order to achieve fairness, it is important to deal with bias in data, algorithmic and user evaluation.

In order to address *Big Data Management*, standard data integration architectures turned out to be inadequate, mainly because of their inability to store and process efficiently large volumes of complex data, novel data integration architectures have been developed. The two most successful ones include a polystore and lambda. Nonetheless, they have not provided solutions to all problems. Still open research and technological problems for such architectures include the development of metadata standards, query optimization techniques, and performance optimization of data integration processes.

Graph embeddings represents a powerful tool for conciliating the representational power of data graphs with the powerful analytics of ML. This approach has been however criticized for creating an additional level of complexity that isolates the input and the output data by a black-box procedure that hamper the explainability of ML results. A promising approach is linking the layers of a neural network with symbols of a *lingua franca* [37]. In this context, Knowledge Graphs could play a crucial role in supporting explainability especially with embeddings methods that exploit the layer of the neural network for generating the vector space.

Fog Computing is a cloud technology in which terminal-generated data does not load directly into the cloud, but is instead pre-processed by decentralized mini data centers. With the advent of Fog Computing, it is possible to perform data processing and data management in the cloud as well as at the edge of the network. As a consequence, it becomes crucial to understand how data management affects the results of complex and distributed processing analytics.

The integration of *relational and NoSQL databases* can be done in many different ways. We used an architecture using a multi-model and multilevel approach with a special abstract model based on typed functional objects. Then a typed lambda calculus can be used as a powerful tool for querying in such integrated database architecture.

References

1. Aberer, K., Boyarsky, A., Cudré-Mauroux, P., Demartini, G., Ruchayskiy, O.: Sciencewise: a web-based interactive semantic platform for scientific collaboration. In: International Semantic Web Conference (ISWC) (2011)
2. Aberer, K., et al.: Emergent semantics principles and issues. In: Lee, Y.J., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 25–38. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24571-1_2
3. Ali, S.M.F., Wrembel, R.: Towards a cost model to optimize user-defined functions in an ETL workflow based on user-defined performance metrics. In: European Conference on Advances in Databases and Information Systems (ADBIS), pp. 441–456 (2019)
4. Allab, K., Labiod, L., Nadif, M.: A semi-NMF-PCA unified framework for data clustering. IEEE Trans. Knowl. Data Eng. (TKDE) **29**(1), 2–16 (2016)
5. Alotaibi, R., Bursztyn, D., Deutsch, A., Manolescu, I., Zampetakis, S.: Towards scalable hybrid stores: constraint-based rewriting to the rescue. In: International Conference on Management of Data (SIGMOD), pp. 1660–1677 (2019)
6. Anderson, W.N., Jr., Morley, T.D.: Eigenvalues of the Laplacian of a graph. Linear Multilinear Algebra **18**(2), 141–145 (1985)
7. Angwin, J., Larson, J., Mattu, S., Kirchner, L.: Machine bias. ProPublica (2016)
8. Barbieri, N., Bonchi, F., Manco, G.: Who to follow and why: link prediction with explanations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1266–1275 (2014)
9. Batini, C., Scannapieco, M.: Data and Information Quality. DSA, Springer, Cham (2016). https://doi.org/10.1007/978-3-319-24106-7
10. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–591 (2002)
11. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Conference on Advances in Neural Information Processing Systems (NIPS), pp. 2787–2795 (2013)
12. Bouguettaya, A., Benatallah, B., Elmargamid, A.: Interconnecting Heterogeneous Information Systems. Kluwer (1998)
13. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv:1312.6203 (2013)

14. Bugiotti, F., Bursztyn, D., Deutsch, A., Manolescu, I., Zampetakis, S.: Flexible hybrid stores: constraint-based rewriting to the rescue. In: IEEE International Conference on Data Engineering (ICDE), pp. 1394–1397 (2016)
15. Cai, D., He, X., Han, J.: Spectral regression: a unified subspace learning framework for content-based image retrieval. In: ACM Multimedia, pp. 403–412 (2007)
16. Cao, S., Lu, W., Xu, Q.: GraRep: Learning graph representations with global structural information. In: International Conference on Information and Knowledge Management (CIKM), pp. 891–900 (2015)
17. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: AAAI Conference on Artificial Intelligence (2016)
18. Catarci, T., Scannapieco, M., Console, M., Demetrescu, C.: My (fair) big data. In: IEEE International Conference on Big Data, pp. 2974–2979 (2017)
19. Ceravolo, P., Zavatarelli, F.: Knowledge acquisition in process intelligence. In: International Conference on Information and Communication Technology Research (ICTRC), pp. 218–221 (2015)
20. Ceravolo, P., et al.: Big data semantics. J. Data Seman. **7**(2), 65–85 (2018)
21. Ceravolo, P., Damiani, E., Torabi, M., Barbon, S.: Toward a new generation of log pre-processing methods for process mining. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNBIP, vol. 297, pp. 55–70. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65015-9_4
22. Ceravolo, P., Guetl, C., Rinderle-Ma, S. (eds.): SIMPDA 2016. LNBIP, vol. 307. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74161-1
23. da Costa, V.G.T., de Leon Ferreira, A.C.P., Junior, S.B., et al.: Strict very fast decision tree: a memory conservative algorithm for data stream mining. Pattern Recogn. Lett. **116**, 22–28 (2018)
24. Cudré-Mauroux, P.: Leveraging knowledge graphs for big data integration: the XI pipeline. Seman. Web **11**(1), 13–17 (2020)
25. Damiani, E., Ardagna, C., Ceravolo, P., Scarabottolo, N.: Toward model-based big data-as-a-service: the TOREADOR approach. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) ADBIS 2017. LNCS, vol. 10509, pp. 3–9. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66917-5_1
26. Dastjerdi, A.V., Buyya, R.: Fog computing: helping the internet of things realize its potential. IEEE Comput. **49**(8), 112–116 (2016)
27. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: ontology based access to distributed and semi-structured information. In: Meersman, R., Tari, Z., Stevens, S. (eds.) Database Semantics. ITIFIP, vol. 11, pp. 351–369. Springer, Boston (1999). https://doi.org/10.1007/978-0-387-35561-0_20
28. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Conference on Advances in Neural Information Processing Systems (NIPS), pp. 3844–3852 (2016)
29. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: Large-scale linked data integration using probabilistic reasoning and crowdsourcing. VLDB J. **22**(5), 665–687 (2013)
30. Duggan, J., et al.: The BigDAWG polystore system. SIGMOD Rec. **44**(2), 11–16 (2015)
31. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.S.: Fairness through awareness. In: Innovations in Theoretical Computer Science, pp. 214–226 (2012)
32. Elmagarmid, A., Rusinkiewicz, M., Sheth, A. (eds.): Management of Heterogeneous and Autonomous Database Systems. Morgan Kaufmann (1999)

33. van Engelen, J.E., Boekhout, H.D., Takes, F.W.: Explainable and efficient link prediction in real-world network data. In: Boström, H., Knobbe, A., Soares, C., Papapetrou, P. (eds.) IDA 2016. LNCS, vol. 9897, pp. 295–307. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46349-0_26
34. Esteves, D., Rula, A., Reddy, A.J., Lehmann, J.: Toward veracity assessment in RDF knowledge bases: an exploratory analysis. J. Data Inf. Qual. 9(3), 16:1–16:26 (2018)
35. Freeman, L.C.: Visualizing social networks. J. Soc. Struct. **1**(1), 4 (2000)
36. Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Díaz, A., Caballero-Mota, Y.: Online and non-parametric drift detection methods based on Hoeffding's bounds. IEEE Trans. Knowl. Data Eng. (TKDE) **27**(3), 810–823 (2014)
37. Futia, G., Vetrò, A.: On the integration of knowledge graphs into deep learning models for a more comprehensible AI? Three challenges for future research. Information **11**(2), 122 (2020)
38. Gadepally, V., et al.: The BigDAWG polystore system and architecture. In: IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6 (2016)
39. Gama, J., Gaber, M.M.: Learning from Data Streams: Processing Techniques in Sensor Networks. Springer, Berlin (2007). https://doi.org/10.1007/3-540-73679-4
40. Gaspar, D., Coric, I. (eds.): Bridging relational and NoSQL databases. In: IGI (2017)
41. Gray, P., Kerschberg, L., King, P., Poulovassilje, A. (eds.): The Functional Approach to Data Management, Modeling, Analyzing and Integrating Heterogeneous Data. Springer, Berlin (2004). https://doi.org/10.1007/978-3-662-05372-0
42. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 855–864 (2016)
43. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Conference on Advances in Neural Information Processing Systems (NIPS), pp. 1024–1034 (2017)
44. Hassan, N., Li, C., Yang, J., Yu, C.: Introduction to the special issue on combating digital misinformation and disinformation. J. Data Inf. Qual. **11**(3), 9:1–9:3 (2019)
45. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv:1506.05163 (2015)
46. Hießl, T., Hochreiner, C., Schulte, S.: Towards a framework for data stream processing in the fog. Informatik Spektrum **42**(4), 256–265 (2019). https://doi.org/10.1007/s00287-019-01192-z
47. Hofmann, T., Buhmann, J.: Multidimensional scaling and data clustering. In: Advances in Neural Information Processing Systems, pp. 459–466 (1995)
48. Hsiao, D.K., Neuhold, E.J., Sacks-Davis, R.: IFIP TC2 WG2.6 Database Semantics Conference on Interoperable Database Systems. Elsevier (2014)
49. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, Berlin (2003). https://doi.org/10.1007/978-3-662-05153-5
50. Jeffery, K.G.: Metadata: the future of information systems. State of the art and research themes, information systems engineering (2000)
51. Jin, X., Wah, B.W., Cheng, X., Wang, Y.: Significance and challenges of big data research. Big Data Res. **2**(2), 59–64 (2015)
52. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Incremental consolidation of data-intensive multi-flows. IEEE Trans. Knowl. Data Eng. (TKDE) **28**(5), 1203–1216 (2016)

53. Jozashoori, S., Vidal, M.: Mapsdi: a scaled-up semantic data integration framework for knowledge graph creation. In: International Conference on the Move to Meaningful Internet Systems (OTM), LNCS, vol. 11877, pp. 58–75 (2019)
54. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)
55. Kolev, B., Bondiombouy, C., Valduriez, P., Jiménez-Peris, R., Pau, R., Pereira, J.: The CloudMdsQL multistore system. In: International Conference on Management of Data (SIGMOD), pp. 2113–2116 (2016)
56. Kuo, T.T., Kim, H.E., Ohno-Machado, L.: Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inform. Assoc. **24**(6), 1211–1220 (2017)
57. Laborie, S., Manzat, A.M., Sèdes, F.: Managing and querying efficiently distributed semantic multimedia metadata collections. IEEE MultiMedia **16**(4), 12–20 (2009)
58. Lara-Benítez, P., Carranza-García, M., García-Gutiérrez, J., Riquelme, J.C.: Asynchronous dual-pipeline deep learning framework for online data stream classification. Integr. Comput. Aided Eng. **1**(2), 1–19 (2020)
59. Lawrence, R.: Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB. In: IEEE International Conference on Computational Science and Computational Intelligence (CSCI), pp. 285–219 (2014)
60. Leida, M., Ceravolo, P., Damiani, E., Asal, R., Colombo, M.: Dynamic access control to semantics-aware streamed process logs. J. Data Seman. **8**(3), 203–218 (2019)
61. Li, S., Da Xu, L., Zhao, S.: 5G internet of things: a survey. J. Ind. Inf. Integr. **10**, 1–9 (2018)
62. Li, X., Dong, X.L., Lyons, K., Meng, W., Srivastava, D.: Truth finding on the deep web: is the problem solved? VLDB Endownment **6**(2), 97–108 (2012)
63. Lin, Y., Liu, Z., Sun, M.: Knowledge representation learning with entities, attributes and relations. Ethnicity **1**, 41–52 (2016)
64. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI Conference on Artificial Intelligence (2015)
65. Mavlyutov, R., Curino, C., Asipov, B., Cudré-Mauroux, P.: Dependency-driven analytics: a compass for uncharted data oceans. In: Conference on Innovative Data Systems Research (CIDR) (2017)
66. Mayer-Schonberger, V., Cukier, K.: Big Data: A Revolution That Will Transform How We Live, Work, and Think. John Murray (2013)
67. Meersman, R., Tari, Z., Stevens, S. (eds.): Database Semantics. ITIFIP, vol. 11. Springer, Boston (1999). https://doi.org/10.1007/978-0-387-35561-0
68. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. CoRR abs/1908.09635 (2019)
69. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M.: Deep learning for IoT big data and streaming analytics: a survey. IEEE Commun. Surv. Tutorials **20**(4), 2923–2960 (2018)
70. Nadal, S., et al.: A software reference architecture for semantic-aware big data systems. Inf. Softw. Technol. (IST) **90**, 75–92 (2017)
71. Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM **62**(8), 36–43 (2019)

72. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1105–1114 (2016)
73. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 701–710 (2014)
74. Perozzi, B., Kulkarni, V., Chen, H., Skiena, S.: Don't walk, skip! online learning of multi-scale network embeddings. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 258–265 (2017)
75. Poggi, A., Rodriguez-Muro, M., Ruzzi, M.: Ontology-based database access with DIG-Mastro and the OBDA plugin for protégé. In: OWLED Workshop on OWL (2008)
76. Pokorný, J.: Database semantics in heterogeneous environment. In: Seminar on Current Trends in Theory and Practice of Informatics (SOFSEM), pp. 125–142 (1996)
77. Pokorný, J.: Functional querying in graph databases. Vietnam J. Comput. Sci. **5**(2), 95–105 (2017)
78. Pokorný, J.: Integration of relational and NoSQL databases. In: Asian Conference on Intelligent Information and Database Systems (ACIIDS), pp. 35–45 (2018)
79. Pokorný, J.: Integration of relational and graph databases functionally. Found. Comput. Decis. Sci. **44**(4), 427–441 (2019)
80. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: International Conference on World Wide Web (WWW), pp. 771–780 (2010)
81. Prokofyev, R., Demartini, G., Cudré-Mauroux, P.: Effective named entity recognition for idiosyncratic web collections. In: International Conference on World Wide Web (WWW), pp. 397–408 (2014)
82. Prokofyev, R., Tonon, A., Luggen, M., Vouilloz, L., Difallah, D.E., Cudré-Mauroux, P.: SANAPHOR: ontology-based coreference resolution. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 458–473. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25007-6_27
83. Qodseya, M.: Visual non-verbal social cues data modeling. In: Woo, C., Lu, J., Li, Z., Ling, T.W., Li, G., Lee, M.L. (eds.) ER 2018. LNCS, vol. 11158, pp. 82–87. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01391-2_16
84. Russom, P.: Data lakes: purposes, practices, patterns, and platforms. TDWI white paper (2017)
85. Scannapieco, M., Batini, C.: Completeness in the relational model: a comprehensive framework. In: International Conference on Information Quality (ICIQ), pp. 333–345 (2004)
86. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
87. Sequeda, J.F., Miranker, D.P.: A pay-as-you-go methodology for ontology-based data access. IEEE Internet Comput. **21**(2), 92–96 (2017)
88. Sequeda, J.F., Briggs, W.J., Miranker, D.P., Heideman, W.P.: A pay-as-you-go methodology to design and build enterprise knowledge graphs from relational databases. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 526–545. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_32
89. Simitsis, A., Vassiliadis, P., Sellis, T.K.: State-space optimization of ETL workflows. IEEE Trans. Knowl. Data Eng. (TKDE) **17**(10), 1404–1419 (2005)
90. Smirnova, A., Audiffren, J., Cudre-Mauroux, P.: APCNN: tackling class imbalance in relation extraction through aggregated piecewise convolutional neural networks. In: Swiss Conference on Data Science (SDS), pp. 63–68 (2019)

91. Smirnova, A., Cudré-Mauroux, P.: Relation extraction using distant supervision: a survey. ACM Comput. Surv. **51**(5), 106:1–106:35 (2018)
92. Souza, A.: Lambda architecture - how to build a big data pipeline (2019). https://towardsdatascience.com
93. Spaccapietra, S., Maryanski, F. (eds.): Data Mining and Reverse Engineering. ITIFIP, Springer, Boston (1998). https://doi.org/10.1007/978-0-387-35300-5
94. Stanchev, P.L., Smeulders, A.W., Groen, F.C.: An approach to image indexing of documents. In: IFIP TC2/WG 2.6 Working Conference on Visual Database Systems, pp. 63–77 (1991)
95. Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T., Hovy, E.: Spine: sparse interpretable neural embeddings. In: AAAI Conference on Artificial Intelligence (2018)
96. Tan, R., Chirkova, R., Gadepally, V., Mattson, T.G.: Enabling query processing across heterogeneous data models: a survey. In: IEEE International Conference on Big Data, pp. 3211–3220 (2017)
97. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: International Conference on World Wide Web (WWW), pp. 1067–1077 (2015)
98. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Disc. **23**(3), 447–478 (2011)
99. Tennant, M., Stahl, F., Rana, O., Gomes, J.B.: Scalable real-time classification of data streams with concept drift. Future Gener. Comput. Syst. **75**, 187–199 (2017)
100. Terrizzano, I., Schwarz, P., Roth, M., Colino, J.E.: Data wrangling: the challenging journey from the wild to the lake. In: Conference on Innovative Data Systems Research (CIDR) (2015)
101. Theocharidis, A., Van Dongen, S., Enright, A.J., Freeman, T.C.: Network visualization and analysis of gene expression data using BioLayout express 3D. Nature Protocols **4**(10), 1535 (2009)
102. Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P., Aberer, K.: *TRank*: ranking entity types using the web of data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 640–656. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_40
103. Tonon, A., Catasta, M., Prokofyev, R., Demartini, G., Aberer, K., Cudre-Mauroux, P.: Contextualized ranking of entity types based on knowledge graphs. J. Web Seman. **37–38**, 170–183 (2016)
104. Tonon, A., Cudré-Mauroux, P., Blarer, A., Lenders, V., Motik, B.: `ArmaTweet`: detecting events by semantic tweet analysis. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10250, pp. 138–153. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58451-5_10
105. Tonon, A., Demartini, G., Cudré-Mauroux, P.: Combining inverted indices and structured search for ad-hoc object retrieval. In: Conference on Research and Development in Information Retrieval, pp. 125–134 (2012)
106. Vaisman, A.A., Zimányi, E.: Data Warehouse Systems - Design and Implementation. Data-Centric Systems and Applications, Springer, Berlin (2014)
107. Valencia-Parra, Á., Varela-Vaca, Á.J., López, M.T.G., Ceravolo, P.: CHAMALEON: framework to improve data wrangling with complex data. In: International Conference on Information Systems (ICIS) (2019)
108. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **38**(1), 49–95 (1996)

109. Vogt, M., Stiemer, A., Schuldt, H.: Polypheny-DB: towards a distributed and self-adaptive polystore. In: IEEE International Conference on Big Data, pp. 3364–3373 (2018)
110. Vyawahare, H., Karde, P.P., Thakare, V.: A hybrid database approach using graph and relational database. In: IEEE International Conference on Research in Intelligent and Computing in Engineering (RICE), pp. 1–4 (2018)
111. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1225–1234 (2016)
112. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. J. Manage. Inf. Syst. **12**(4), 5–33 (1996)
113. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI Conference on Artificial Intelligence (2014)
114. Weinberger, K.Q., Sha, F., Saul, L.K.: Learning a kernel matrix for nonlinear dimensionality reduction. In: International Conference on Machine Learning (ICML), p. 106 (2004)
115. Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Comput. **25**(3), 38–49 (1992)
116. Wrembel, R., Abelló, A., Song, I.: DOLAP data warehouse research over two decades: trends and challenges. Inf. Syst. **85**, 44–47 (2019)
117. Xie, Q., Ma, X., Dai, Z., Hovy, E.: An interpretable knowledge transfer model for knowledge base completion. arXiv:1704.05908 (2017)
118. Yamamoto, S., Mori, H. (eds.): HIMI 2018. LNCS, vol. 10905. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92046-7
119. Yang, C., Sun, M., Liu, Z., Tu, C.: Fast network embedding enhancement via high order proximity approximation. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 3894–3900 (2017)
120. Yue, X., et al.: Graph embedding on biomedical networks: methods, applications and evaluations. Bioinformatics **36**(4), 1241–1251 (2020)
121. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 353–362 (2016)