Journal of Software Engineering and Applications, 2015, 8, 617-634
Published Online December 2015 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2015.812058

Data Modeling and Data Analytics: A Survey from a Big Data Perspective

André Ribeiro, Afonso Silva, Alberto Rodrigues da Silva

INESC-ID/Instituto Superior Técnico, Lisbon, Portugal
Email: andre.ribeiro@tecnico.ulisboa.pt, afonso.silva@tecnico.ulisboa.pt, alberto.silva@tecnico.ulisboa.pt

Received 20 October 2015; accepted 27 December 2015; published 30 December 2015

Abstract

These last years we have been witnessing a tremendous growth in the volume and availability of data. This fact results primarily from the emergence of a multitude of sources (e.g. computers, mobile devices, sensors or social networks) that are continuously producing either structured, semi-structured or unstructured data. Database Management Systems and Data Warehouses are no longer the only technologies used to store and analyze datasets, namely due to the volume and complex structure of nowadays data that degrade their performance and scalability. Big Data is one of the recent challenges, since it implies new requirements in terms of data storage, processing and visualization. Despite that, analyzing properly Big Data can constitute great advantages because it allows discovering patterns and correlations in datasets. Users can use this processed information to gain deeper insights and to get business advantages. Thus, data modeling and data analytics are evolved in a way that we are able to process huge amounts of data without compromising performance and availability, but instead by "relaxing" the usual ACID properties. This paper provides a broad view and discussion of the current state of this subject with a particular focus on data modeling and data analytics, describing and clarifying the main differences between the three main approaches in what concerns these aspects, namely: operational databases, decision support databases and Big Data technologies.

Keywords

Data Modeling, Data Analytics, Modeling Language, Big Data

1. Introduction

We have been witnessing to an exponential growth of the volume of data produced and stored. This can be explained by the evolution of the technology that results in the proliferation of data with different formats from the most various domains (e.g. health care, banking, government or logistics) and sources (e.g. sensors, social networks or mobile devices). We have assisted a paradigm shift from simple books to sophisticated databases that keep being populated every second at an immensely fast rate. Internet and social media also highly contribute to the worsening of this situation [1]. Facebook, for example, has an average of 4.75 billion pieces of content shared among friends every day [2]. Traditional Relational Database Management Systems (RDBMSs) and Data Warehouses (DWs) are designed to handle a certain amount of data, typically structured, which is completely different from the reality that we are facing nowadays. Business is generating enormous quantities of data that are too big to be processed and analyzed by the traditional RDBMSs and DWs technologies, which are struggling to meet the performance and scalability requirements.

Therefore, in the recent years, a new approach that aims to mitigate these limitations has emerged. Companies like Facebook, Google, Yahoo and Amazon are the pioneers in creating solutions to deal with these "Big Data" scenarios, namely recurring to technologies like Hadoop [3] [4] and MapReduce [5]. Big Data is a generic term used to refer to massive and complex datasets, which are made of a variety of data structures (structured, semi-structured and unstructured data) from a multitude of sources [6]. Big Data can be characterized by three Vs: volume (amount of data), velocity (speed of data in and out) and variety (kinds of data types and sources) [7]. Still, there are added some other Vs for variability, veracity and value [8].

Adopting Big Data-based technologies not only mitigates the problems presented above, but also opens new perspectives that allow extracting value from Big Data. Big Data-based technologies are being applied with success in multiple scenarios [1] [9] [10] like in: (1) e-commerce and marketing, where count the clicks that the crowds do on the web allow identifying trends that improve campaigns, evaluate personal profiles of a user, so that the content shown is the one he will most likely enjoy; (2) government and public health, allowing the detection and tracking of disease outbreaks via social media or detect frauds; (3) transportation, industry and surveillance, with real-time improved estimated times of arrival and smart use of resources.

This paper provides a broad view of the current state of this area based on two dimensions or perspectives: Data Modeling and Data Analytics. **Table 1** summarizes the focus of this paper, namely by identifying three representative approaches considered to explain the evolution of Data Modeling and Data Analytics. These approaches are: Operational databases, Decision Support databases and Big Data technologies.

This research work has been conducted in the scope of the DataStorm project [11], led by our research group, which focuses on addressing the design, implementation and operation of the current problems with Big Data-based applications. More specifically, the goal of our team in this project is to identify the main concepts and patterns that characterize such applications, in order to define and apply suitable domain-specific languages (DSLs). Then these DSLs will be used in a Model-Driven Engineering (MDE) [12]-[14] approach aiming to ease the design, implementation and operation of such data-intensive applications.

To ease the explanation and better support the discussion throughout the paper, we use a very simple case study based on a fictions academic management system described below:

Case Study—Academic Management System (AMS):

The Academic Management System (AMS) should support two types of end-users: students and professors. Each person has a name, gender, date of birth, ID card, place of origin and country. Students are enrolled in a given academic program, which is composed of many courses. Professors have an academic degree, are associated to a given department and lecture one or more courses. Each course has a name, academic term and can have one or more locations and academic programs associated. Additionally, a course is associated to a schedule composed of many class periods determining its duration and the day it occurs.

The outline of this paper is as follows: Section 2 describes Data Modeling and some representative types of data models used in operational databases, decision support databases and Big Data technologies. Section 3 details the type of operations performed in terms of Data Analytics for these three approaches. Section 4 compares and discusses each approach in terms of the Data Modeling and Data Analytics perspectives. Section 5 discusses our research in comparison with the related work. Finally, Section 6 concludes the paper by summarizing its key points and identifying future work.

2. Data Modeling

This section gives an in-depth look of the most popular data models used to define and support Operational Databases, Data Warehouses and Big Data technologies.

Table 1. Approaches and perspectives of the survey.

Approaches	Operational	Decision Support	Big Data
Data Modeling Perspective	ER and Relational Models	Star Schema and OLAP Cube Models	Key-Value, Document, Wide-Column and Graph
	RDBMS	DW	Big Data-Based Systems
Data Analytics Perspective	OLTP	OLAP	Multiple Classes (Batch-oriented processing, stream-processing, OLTP and Interactive ad-hoc queries)

Databases are widely used either for personal or enterprise use, namely due to their strong ACID guarantees (atomicity, consistency, isolation and durability) guarantees and the maturity level of Database Management Systems (DBMSs) that support them [15].

The data modeling process may involve the definition of three data models (or schemas) defined at different abstraction levels, namely Conceptual, Logical and Physical data models [15] [16]. **Figure 1** shows part of the three data models for the AMS case study. All these models define three entities (Person, Student and Professor) and their main relationships (teach and supervise associations).

Conceptual Data Model. A conceptual data model is used to define, at a very high and platform-independent level of abstraction, the entities or concepts, which represent the data of the problem domain, and their relationships. It leaves further details about the entities (such as their attributes, types or primary keys) for the next steps. This model is typically used to explore domain concepts with the stakeholders and can be omitted or used instead of the logical data model.

Logical Data Model. A logical data model is a refinement of the previous conceptual model. It details the domain entities and their relationships, but standing also at a platform-independent level. It depicts all the attributes that characterize each entity (possibly also including its unique identifier, the primary key) and all the relationships between the entities (possibly including the keys identifying those relationships, the foreign keys). Despite being independent of any DBMS, this model can easily be mapped on to a physical data model thanks to the details it provides.

Physical Data Model. A physical data model visually represents the structure of the data as implemented by a given class of DBMS. Therefore, entities are represented as tables, attributes are represented as table columns and have a given data type that can vary according to the chosen DBMS, and the relationships between each table are identified through foreign keys. Unlike the previous models, this model tends to be platform-specific, because it reflects the database schema and, consequently, some platform-specific aspects (e.g. database-specific data types or query language extensions).

Summarizing, the complexity and detail increase from a conceptual to a physical data model. First, it is important to perceive at a higher level of abstraction, the data entities and their relationships using a Conceptual Data Model. Then, the focus is on detailing those entities without worrying about implementation details using a Logical Data Model. Finally, a Physical Data Model allows to represent how data is supported by a given DBMS [15] [16].

2.1. Operational Databases

Databases had a great boost with the popularity of the Relational Model [17] proposed by E. F. Codd in 1970. The Relational Model overcame the problems of predecessors data models (namely the Hierarchical Model and the Navigational Model [18]). The Relational Model caused the emergence of Relational Database Management Systems (RDBMSs), which are the most used and popular DBMSs, as well as the definition of the Structured Query Language (SQL) [19] as the standard language for defining and manipulating data in RDBMSs. RDBMSs are widely used for maintaining data of daily operations. Considering the data modeling of operational databases there are two main models: the Relational and the Entity-Relationship (ER) models.

Relational Model. The Relational Model is based on the mathematical concept of relation. A relation is defined as a set (in mathematics terminology) and is represented as a table, which is a matrix of columns and rows, holding information about the domain entities and the relationships among them. Each column of the table corresponds to an entity attribute and specifies the attribute's name and its type (known as domain). Each row of

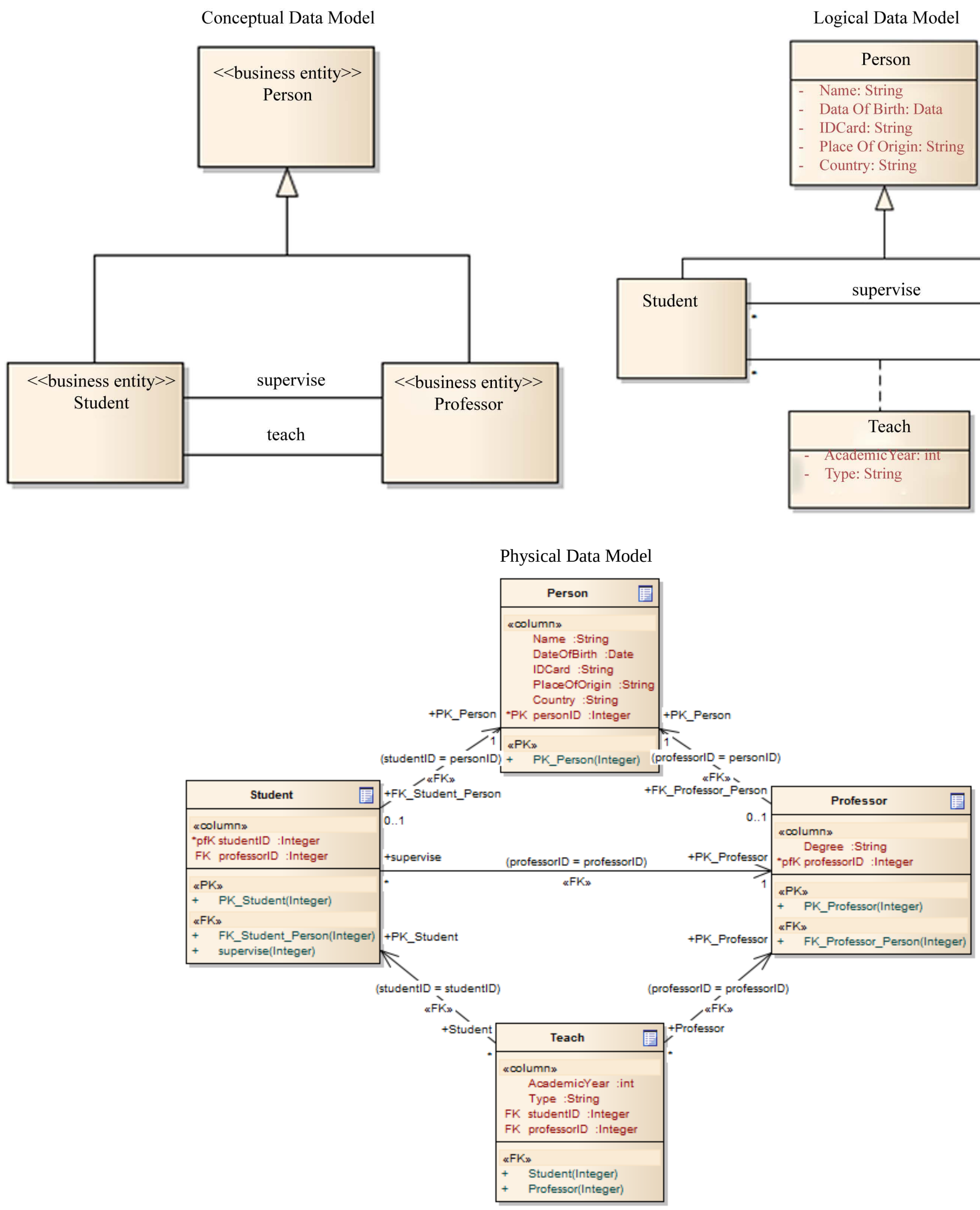


Figure 1. Example of three data models (at different abstraction levels) for the Academic Management System.

the table (known as tuple) corresponds to a single element of the represented domain entity. In the Relational Model each row is unique and therefore a table has an attribute or set of attributes known as **primary key**, used to univocally identify those rows. Tables are related with each other by sharing one or more common attributes. These attributes correspond to a primary key in the referenced (parent) table and are known as **foreign keys** in the referencing (child) table. In **one-to-many relationships**, the referenced table corresponds to the entity of the "one" side of the relationship and the referencing table corresponds to the entity of the "many" side. In **many-to-many relationships**, it is used an additional association table that associates the entities involved through their respective primary keys. The Relational Model also features the concept of **View**, which is like a table whose rows are not explicitly stored in the database, but are computed as needed from a view definition. Instead, a view is defined as a query on one or more base tables or other views [17].

Entity-Relationship (ER) Model. The Entity Relationship (ER) Model [20], proposed by Chen in 1976, appeared as an alternative to the Relational Model in order to provide more expressiveness and semantics into the database design from the user's point of view. The ER model is a semantic data model, *i.e*. aims to represent the meaning of the data involved on some specific domain. This model was originally defined by three main concepts: entities, relationships and attributes. An **entity** corresponds to an object in the real world that is distinguishable from all other objects and is characterized by a set of attributes. Each **attribute** has a range of possible values, known as its domain, and each entity has its own value for each attribute. Similarly to the Relational Model, the set of attributes that identify an entity is known as its primary key.

Entities can be though as **nouns** and correspond to the tables of the Relational Model. In turn, a **relationship** is an association established among two or more entities. A relationship can be thought as a *verb* and includes the roles of each participating entities with multiplicity constraints, and their cardinality. For instance, a relationship can be of one-to-one (1:1), one-to-many (1:M) or many-to-many (M:N). In an ER diagram, entities are usually represented as rectangles, attributes as circles connected to entities or relationships through a line, and relationships as diamonds connected to the intervening entities through a line.

The Enhanced ER Model [21] provided additional concepts to represent more complex requirements, such as generalization, specialization, aggregation and composition. Other popular variants of ER diagram notations are Crow's foot, Bachman, Barker's, IDEF1X and UML Profile for Data Modeling [22].

2.2. Decision Support Databases

The evolution of relational databases to decision support databases, hereinafter indistinctly referred as "**Data Warehouses**" **(DWs)**, occurred with the need of storing operational but also historical data, and the need of analyzing that data in complex dashboards and reports. Even though a DW seems to be a relational database, it is different in the sense that DWs are more suitable for supporting query and analysis operations (fast reads) instead of transaction processing (fast reads and writes) operations. DWs contain historical data that come from transactional data, but they also might include other data sources [23]. DWs are mainly used for **OLAP (online analytical processing)** operations. OLAP is the approach to provide report data from DW through multi-dimensional queries and it is required to create a multi-dimensional database [24].

Usually, DWs include a framework that allows extracting data from multiple data sources and transform it before loading to the repository, which is known as ETL (Extract Transform Load) framework [23].

Data modeling in DW consists in defining fact tables with several dimension tables, suggesting **star or snowflake schema** data models [23]. A star schema has a central **fact table** linked with **dimension tables**. Usually, a fact table has a large number of attributes (in many cases in a denormalized way), with many foreign keys that are the primary keys to the dimension tables. The dimension tables represent characteristics that describe the fact table. When star schemas become too complex to be queried efficiently they are transformed into multi-dimensional arrays of data called **OLAP cubes** (for more information on how this transformation is performed the reader can consult the following references [24] [25]).

A star schema is transformed to a cube by putting the fact table on the front face that we are facing and the dimensions on the other faces of the cube [24]. For this reason, cubes can be equivalent to star schemas in content, but they are accessed with more platform-specific languages than SQL that have more analytic capabilities (e.g. MDX or XMLA). A cube with three dimensions is conceptually easier to visualize and understand, but the OLAP cube model supports more than three dimensions, and is called a hypercube.

Figure 2 shows two examples of star schemas regarding the case study AMS. The star schema on the left represents the data model for the Student's fact, while the data model on the right represents the Professor's fact. Both of them have a central fact table that contains specific attributes of the entity in analysis and also foreign keys to the dimension tables. For example, a Student has a place of origin (DIM_PLACEOFORIGIN) that is described by a city and associated to a country (DIM_COUNTRY) that has a name and an ISO code. On the other hand, **Figure 3** shows a cube model with three dimensions for the Student. These dimensions are represented by sides of the cube (Student, Country and Date). This cube is useful to execute queries such as: the students by country enrolled for the first time in a given year.

A challenge that DWs face is the growth of data, since it affects the number of dimensions and levels in either the star schema or the cube hierarchies. The increasing number of dimensions over time makes the management of such systems often impracticable; this problem becomes even more serious when dealing with Big Data scenarios, where data is continuously being generated [23].

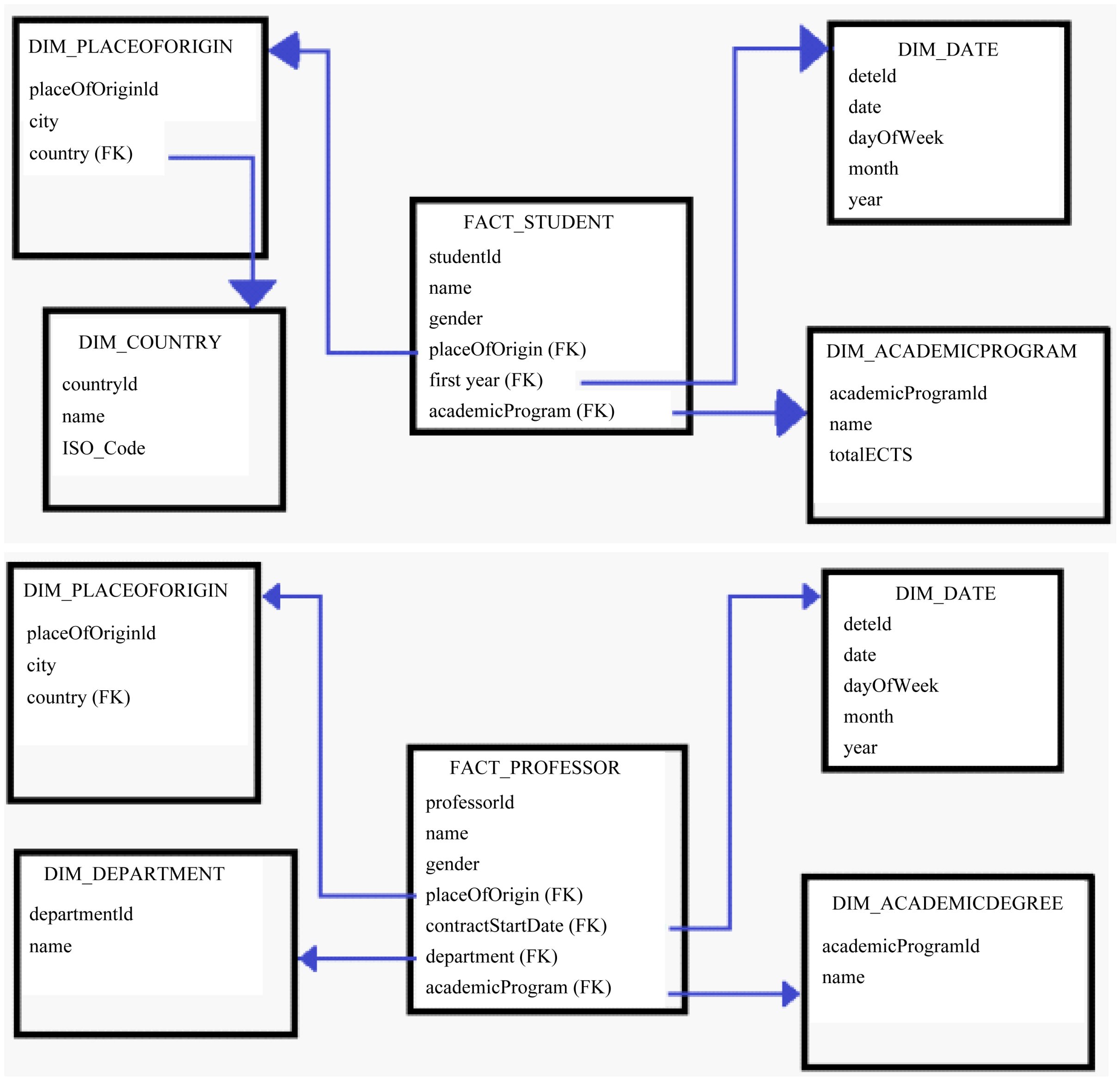


Figure 2. Example of two star schema models for the Academic Management System.

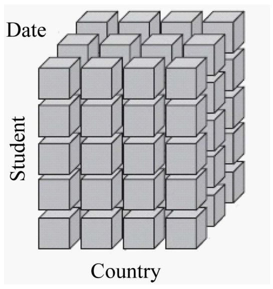


Figure 3. Example of a cube model for the Academic Management System.

2.3. Big Data Technologies

The volume of data has been exponentially increasing over the last years, namely due to the simultaneous growth of the number of sources (e.g. users, systems or sensors) that are continuously producing data. These data sources produce huge amounts of data with variable representations that make their management by the traditional RDBMSs and DWs often impracticable. Therefore, there is a need to devise new data models and technologies that can handle such Big Data.

NoSQL (Not Only SQL) [26] is one of the most popular approaches to deal with this problem. It consists in a group of non-relational DBMSs that consequently do not represent databases using tables and usually do not use SQL for data manipulation. NoSQL systems allow managing and storing large-scale denormalized datasets, and are designed to scale horizontally. They achieve that by compromising consistency in favor of availability and partition-tolerance, according to Brewer's CAP theorem [27]. Therefore, NoSQL systems are "eventually consistent", *i.e*. assume that writes on the data are eventually propagated over time, but there are limited guarantees that different users will read the same value at the same time. NoSQL provides BASE guarantees (Basically Available, Soft state and Eventually consistent) instead of the traditional ACID guarantees, in order to greatly improve performance and scalability [28].

NoSQL databases can be classified in four categories [29]: Key-value stores, (2) Document-oriented databases, (3) Wide-column stores, and (4) Graph databases.

Key-value Stores. A Key-Value store represents data as a collection (known as dictionary or map) of key-value pairs. Every *key* consists in a unique alphanumeric identifier that works like an index, which is used to access a corresponding value. *Values* can be simple text strings or more complex structures like arrays. The Key-value model can be extended to an ordered model whose keys are stored in lexicographical order. The fact of being a simple data model makes Key-value stores ideally suited to retrieve information in a very fast, available and scalable way. For instance, Amazon makes extensive use of a Key-value store system, named Dynamo, to manage the products in its shopping cart [30]. Amazon's Dynamo and Voldemort, which is used by Linkedin, are two examples of systems that apply this data model with success. An example of a key-value store for both students and professors of the Academic Managements System is shown in **Figure 4**.

Document-oriented Databases. Document-oriented databases (or document stores) were originally created to store traditional documents, like a notepad text file or Microsoft Word document. However, their concept of document goes beyond that, and a document can be any kind of domain object [26]. Documents contain encoded data in a standard format like XML, YAML, JSON or BSON (Binary JSON) and are univocally identified in the database by a unique key. Documents contain semi-structured data represented as name-value pairs, which can vary according to the row and can nest other documents. Unlike key-value stores, these systems support secondary indexes and allow fully searching either by keys or values. Document databases are well suited for storing and managing huge collections of textual documents (e.g. text files or email messages), as well as semi-structured or denormalized data that would require an extensive use of "nulls" in an RDBMS [30]. MongoDB and CouchDB are two of the most popular Document-oriented database systems. **Figure 5** illustrates two collections of documents for both students and professors of the Academic Management System.

Students	
Key	Value
1	Name: Jean Grey
	DateOfBirth: 19-05-1963
	IDCard: 1234567
	PlaceOfOrigin: Austin
	Country: USA
	AcademicProgram_ID:1
2	Name: Scott Summers
	DateOfBirth: 12-10-1968
	IDCard: 765414A
	Supervisor: {
	Name: Emma Frost
	DateOfBirth: 1-1-1936
	IDCard: 222222
	}

Professors	
Key	Value
1	Name: Charles Xavier
	DateOfBirth: 13-07-1940
	IDCard: 111111
	PlaceOfOrigin: Mirfield
	Country: UK
2	Name: Emma Frost
	DateOfBirth: 1-1-1936
	IDCard: 222222

Figure 4. Example of a key-value store for the Academic Management System.

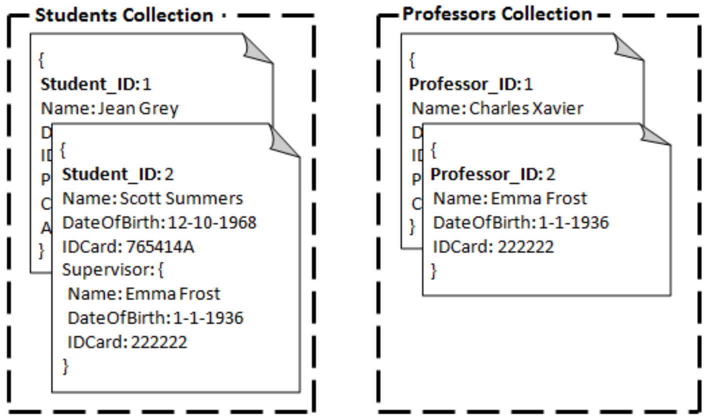


Figure 5. Example of a documents-oriented database for the Academic Management System.

Wide-column Stores. Wide-column stores (also known as column-family stores, extensible record stores or column-oriented databases) represent and manage data as sections of columns rather than rows (like in RDBMS). Each section is composed of key-value pairs, where the keys are rows and the values are sets of columns, known as column families. Each row is identified by a primary key and can have column families different of the other rows. Each column family also acts as a primary key of the set of columns it contains. In turn each column of column family consists in a name-value pair. Column families can even be grouped in super column families [29]. This data model was highly inspired by Google's BigTable [31]. Wide-column stores are suited for scenarios like: (1) Distributed data storage; (2) Large-scale and batch-oriented data processing, using the famous MapReduce method for tasks like sorting, parsing, querying or conversion and; (3) Exploratory and predictive analytics. Cassandra and Hadoop HBase are two popular frameworks of such data management systems [29]. **Figure 6** depicts an example of a wide-column store for the entity "person" of the Academic Managements System.

Graph Databases. Graph databases represent data as a network of nodes (representing the domain entities) that are connected by edges (representing the relationships among them) and are characterized by properties expressed as key-value pairs. Graph databases are quite useful when the focus is on exploring the relationships between data, such as traversing social networks, detecting patterns or infer recommendations. Due to their visual representation, they are more user-friendly than the aforementioned types of NoSQL databases. Neo4j and Allegro Graph are two examples of such systems.

3. Data Analytics

This section presents and discusses the types of operations that can be performed over the data models described in the previous section and also establishes comparisons between them. A complementary discussion is provided in Section 4.

3.1. Operational Databases

Systems using operational databases are designed to handle a high number of transactions that usually perform changes to the operational data, *i.e.* the data an organization needs to assure its everyday normal operation. These systems are called **Online Transaction Processing (OLTP)** systems and they are the reason why RDBMSs are so essential nowadays. RDBMSs have increasingly been optimized to perform well in OLTP systems, namely providing reliable and efficient data processing [16].

The set of operations supported by RDBMSs is derived from the relational algebra and calculus underlying the Relational Model [15]. As mentioned before, SQL is the standard language to perform these operations. SQL can be divided in two parts involving different types of operations: Data Definition Language (SQL-DDL) and Data Manipulation Language (SQL-DML).

SQL-DDL allows performing the creation (CREATE), update (UPDATE) and deletion (DROP) of the vari-

Column Family: Students						
Key				Columns		
1	Name	DateOfBirth	IDCard	PlaceOfOrigin	Country	
	Jean Grey	19-05-1963	1234567	Austin	USA	
2	Name	DateOfBirth	IDCard	Supervisor		
	Scott Summers	12-10-1968	765414A	Name	DateOfBirth	IDCard
				Emma Frost	1-1-1936	222222

Column Family: Professors					
Key				Columns	
1	Name	DateOfBirth	IDCard	PlaceOfOrigin	Country
	Charles Xavier	13-07-1940	111111	Mirfield	UK
2	Name	DateOfBirth	IDCard		
	Emma Frost	1-1-1936	222222		

Figure 6. Example of a wide-column store for the Academic Management System.

ous database objects. First it allows managing schemas, which are named collections of all the database objects that are related to one another. Then inside a schema, it is possible to manage tables specifying their columns and types, primary keys, foreign keys and constraints. It is also possible to manage views, domains and indexes. An index is a structure that speeds up the process of accessing to one or more columns of a given table, possibly improving the performance of queries [15] [16].

For example, considering the Academic Management System, a system manager could create a table for storing information of a student by executing the following SQL-DDL command:

```
CREATE TABLE Student (
        Student ID NOT NULL IDENTITY,
        Name VARCHAR(255) NOT NULL,
        Date of Birth DATE NOT NULL,
        ID Card VARCHAR(255) NOT NULL,
        Place of Origin VARCHAR(255),
        Country VARCHAR(255),
        PRIMARY KEY (Student ID))
```

On the other hand, **SQL-DML** is the language that enables to manipulate database objects and particularly to extract valuable information from the database. The most commonly used and complex operation is the SELECT operation, which allows users to query data from the various tables of a database. It is a powerful operation because it is capable of performing in a single query the equivalent of the relational algebra's selection, projection and join operations. The SELECT operation returns as output a table with the results. With the SELECT operation is simultaneously possible to: define which tables the user wants to query (through the FROM clause), which rows satisfy a particular condition (through the WHERE clause), which columns should appear in the result (through the SELECT clause), order the result (in ascending or descending order) by one or more columns (through the ORDER BY clause), group rows with the same column values (through the GROUP BY clause) and filter those groups based on some condition (through the HAVING clause). The SELECT operation also allows using aggregation functions, which perform arithmetic computation or aggregation of data (e.g. counting or summing the values of one or more columns).

Many times there is the need to combine columns of more than one table in the result. To do that, the user can use the JOIN operation in the query. This operation performs a subset of the Cartesian product between the involved tables, *i.e.* returns the row pairs where the matching columns in each table have the same value. The most common queries that use joins involve tables that have one-to-many relationships. If the user wants to include in the result the rows that did not satisfied the join condition, then he can use the outer joins operations (left, right and full outer join). Besides specifying queries, DML allows modifying the data stored in a database. Namely, it allows adding new rows to a table (through the INSERT statement), modifying the content of a given table's rows (through the UPDATE statement) and deleting rows from a table (through the DELETE statement) [16].

SQL-DML also allows combining the results of two or more queries into a single result table by applying the Union, Intersect and Except operations, based on the Set Theory [15].

For example, considering the Academic Management System, a system manager could get a list of all students who are from G8 countries by entering the following SQL-DML query:

```
SELECT Name, Country
FROM Student
WHERE Country in ("Canada", "France", "Germany", "Italy", "Japan", "Russia", "UK", "USA")
ORDER BY Country
```

3.2. Decision Support Databases

The most common data model used in DW is the OLAP cube, which offers a set of operations to analyze the cube model [23]. Since data is conceptualized as a cube with hierarchical dimensions, its operations have familiar names when manipulating a cube, such as slice, dice, drill and pivot. **Figure 7** depicts these operations considering the Student's facts of the AMS case study (see **Figure 2**).

The **slice** operation begins by selecting one of the dimensions (or faces) of the cube. This dimension is the one we want to consult and it is followed by "slicing" the cube to a specific depth of interest. The slice operation leaves us with a more restricted selection of the cube, namely the dimension we wanted (front face) and the layer of that dimension (the sliced section). In the example of **Figure 7** (top-left), the cube was sliced to consider only data of the year 2004.

Dice is the operation that allows restricting the front face of the cube by reducing its size to a smaller targeted domain. This means that the user produces a smaller "front face" than the one he had at the start. **Figure 7** (top-right) shows that the set of students has decreased after the dice operation.

Drill is the operation that allows to navigate by specifying different levels of the dimensions, ranging from the most detailed ones (**drill down**) to the most summarized ones **(drill up**). **Figure 7** (bottom-left) shows the drill down so the user can see the cities from where the students of the country Portugal come from.

The **pivot** operation allows changing the dimension that is being faced (change the current front face) to one that is adjacent to it by rotating the cube. By doing this, the user obtains another perspective of the data, which requires the queries to have a different structure but can be more beneficial for specific queries. For instance, he can slice and dice the cube away to get the results he needed, but sometimes with a pivot most of those operations can be avoided by perceiving a common structure on future queries and pivoting the cube in the correct fashion [23] [24]. **Figure 7** (bottom-right) shows a pivot operation where years are arranged vertically and countries horizontally.

The usual operations issued over the OLAP cube are about just querying historical events stored in it. So,

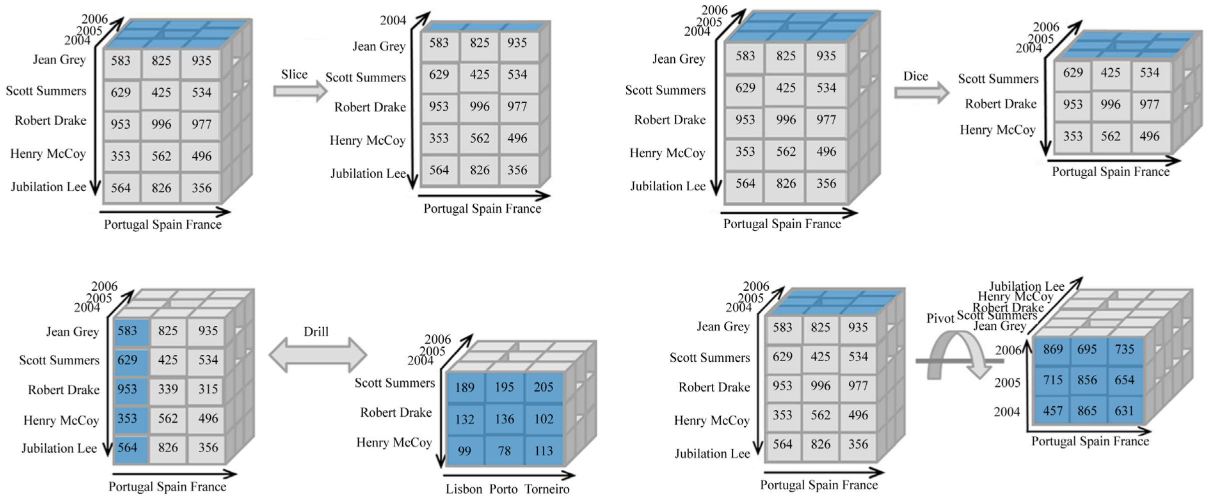


Figure 7. Representation of cube operations for the Academic Management System: slice (top-left), dice (top-right), drill up/down (bottom-left) and pivot (bottom-right).

a common dimension is a dimension associated to time. The most popular language for manipulating OLAP cubes is MDX (Multidimensional Expressions) [32], which is a query language for OLAP databases that supports all the operations mentioned above. MDX is exclusively used to analyze and read data since it was not designed with SQL-DML in mind. The star schema and the OLAP cube are designed *a priori* with a specific purpose in mind and cannot accept queries that differentiate much from the ones they were design to respond too. The benefit in this, is that queries are much simpler and faster, and by using a cube it is even quicker to detect patterns, find trends and navigate around the data while "slicing and dicing" with it [23] [25].

Again considering the Academic Management System example, the following query represents an MDX select statement. The SELECT clause sets the query axes as the name and the gender of the Student dimension and the year 2015 of the Date dimension. The FROM clause indicates the data source, here being the Students cube, and the WHERE clause defines the slicer axis as the "Computer Science" value of the Academic Program dimension. This query returns the students (by names and gender) that have enrolled in Computer Science in the year 2015.

```
SELECT
     { [Student].[Name],
        [Student].[Gender]} ON COLUMNS
     { [Date].[Academic Year] &[2015] } ON ROWS
FROM [Students Cube]
WHERE ([Academic Program].[Name] &[Computer Science])
```

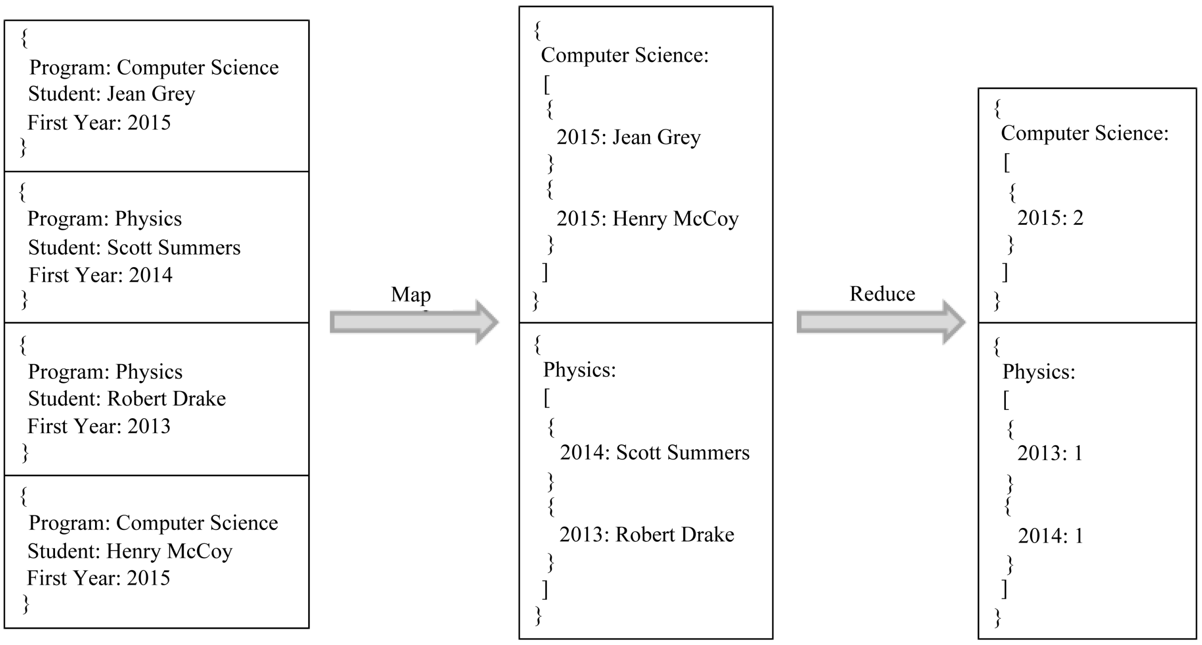
3.3. Big Data Technologies

Big Data Analytics consists in the process of discovering and extracting potentially useful information hidden in huge amounts of data (e.g. discover unknown patterns and correlations). Big Data Analytics can be separated in the following categories: (1) Batch-oriented processing; (2) Stream processing; (3) OLTP and; (4) Interactive ad-hoc queries and analysis.

Batch-oriented processing is a paradigm where a large volume of data is firstly stored and only then analyzed, as opposed to Stream processing. This paradigm is very common to perform large-scale recurring tasks in parallel like parsing, sorting or counting. The most popular batch-oriented processing model is MapReduce [5], and more specifically its open-source implementation in Hadoop[1]. MapReduce is based on the divide and conquer (D&C) paradigm to break down complex Big Data problems into small sub-problems and process them in parallel. MapReduce, as its name hints, comprises two major functions: Map and Reduce. First, data is divided into small chunks and distributed over a network of nodes. Then, the Map function, which performs operations like filtering or sorting, is applied simultaneously to each chunk of data generating intermediate results. After that, those intermediate results are aggregated through the Reduce function in order to compute the final result. **Figure 8** illustrates an example of the application of MapReduce in order to calculate the number of students enrolled in a given academic program by year. This model schedules computation resources close to data location, which avoids the communication overhead of data transmission. It is simple and widely applied in bioinformatics, web mining and machine learning. Also related to Hadoop's environment, Pig[2] and Hive[3] are two frameworks used to express tasks for Big Data sets analysis in MapReduce programs. Pig is suitable for data flow tasks and can produce sequences of MapReduce programs, whereas Hive is more suitable for data summarization, queries and analysis. Both of them use their own SQL-like languages, Pig Latin and Hive QL, respectively [33]. These languages use both CRUD and ETL operations.

Streaming processing is a paradigm where data is continuously arriving in a stream, at real-time, and is analyzed as soon as possible in order to derive approximate results. It relies in the assumption that the potential value of data depends on its freshness. Due to its volume, only a portion of the stream is stored in memory [33]. Streaming processing paradigm is used in online applications that need real-time precision (e.g. dashboards of production lines in a factory, calculation of costs depending on usage and available resources). It is supported by Data Stream Management Systems (DSMS) that allow performing SQL-like queries (e.g. select, join, group, count) within a given window of data. This window establishes the period of time (based on time) or number of events (based on length) [34]. Storm and S4 are two examples of such systems.

[1]https://hadoop.apache.org
[2]https://pig.apache.org
[3]https://hive.apache.org



{ Program: Computer Science Student: Jean Grey First Year: 2015 }
{ Program: Physics Student: Scott Summers First Year: 2014 }
{ Program: Physics Student: Robert Drake First Year: 2013 }
{ Program: Computer Science Student: Henry McCoy First Year: 2015 }
Map
{ Computer Science: [{ 2015: Jean Grey } { 2015: Henry McCoy }] }
{ Physics: [{ 2014: Scott Summers } { 2013: Robert Drake }] }
Reduce
{ Computer Science: [{ 2015: 2 }] }
{ Physics: [{ 2013: 1 } { 2014: 1 }] }

Figure 8. Example of Map Reduce applied to the Academic Management System.

OLTP, as we have seen before, is mainly used in the traditional RDBMS. However, these systems cannot assure an acceptable performance when the volume of data and requests is huge, like in Facebook or Twitter. Therefore, it was necessary to adopt NoSQL databases that allow achieving very high performances in systems with such large loads. Systems like Cassandra[4], HBase[5] or MongoDB[6] are effective solutions currently used. All of them provide their own query languages with equivalent CRUD operations to the ones provided by SQL. For example, in Cassandra is possible to create Column Families using CQL, in HBase is possible to delete a column using Java, and in MongoDB insert a document into a collection using JavaScript. Below there is a query in JavaScript for a MongoDB database equivalent to the SQL-DML query presented previously.

```
db.students.find({ country: [“Canada”, “France”, “Germany”, “Italy”, “Japan”, “Russia”, “UK”, “USA”] }, { name: 1, country: 1 }).
sort({ country: 1 })
```

At last, **Interactive ad-hoc queries and analysis** consists in a paradigm that allows querying different large-scale data sources and query interfaces with a very low latency. This type of systems argue that queries should not need more then few seconds to execute even in a Big Data scale, so that users are able to react to changes if needed. The most popular of these systems is Drill[7]. Drill works as a query layer that transforms a query written in a human-readable syntax (e.g. SQL) into a logical plan (query written in a platform-independent way). Then, the logical plan is transformed into a physical plan (query written in a platform-specific way) that is executed in the desired data sources (e.g. Cassandra, HBase or MongoDB) [35].

4. Discussion

In this section we compare and discuss the approaches presented in the previous sections in terms of the two perspectives that guide this survey: Data Modeling and Data Analytics. Each perspective defines a set of features used to compare Operational Databases, DWs and Big Data approaches among themselves.

Regarding the ***Data Modeling Perspective***, **Table 2** considers the following features of analysis: (1) the data model; (2) the abstraction level in which the data model resides, according to the abstraction levels (Conceptual, Logical and Physical) of the database design process; (3) the concepts or constructs that compose the data model;

[4]http://cassandra.apache.org
[5]https://hbase.apache.org
[6]https://www.mongodb.org
[7]https://drill.apache.org

Table 2. Comparison of the approaches from the Data Modeling Perspective.

Approaches / Features	Data Model	Abstraction Level	Concepts	Concrete Languages	Modeling Tools	DB Tools Support
Operational	Entity-Relationship Model	Conceptual, Logical	Entity Relationship Attribute Primary Key Foreign Key	Chen's, Crow's foot, Bachman's, Barker's, IDEF1X	Sparx Enterprise Architect, Visual Paradigm, Oracle Designer, MySQL Workbench, ER/Studio	
	Relational Model	Logical, Physical	Table Row Attribute Primary Key Foreign Key, View, Index	SQL-DDL, UML Data Profile	Sparx Enterprise Architect, Visual Paradigm, Oracle Designer, MySQL Workbench, ER/Studio	Microsoft SQL Server, Oracle, MySQL, PostgreSQL, IBM DB2
Decision Support	OLAP Cube	Conceptual, Logical	Dimensions, Levels, Cube faces, Time dimension, Local dimension	Common Warehouse Metamodel	Essbase Studio Tool, Enterprise Architect, Visual Paradigm	Oracle Warehouse Builder, Essbase Studio Tool, Microsoft Analysis Services
	Star Schema	Logical, Physical	Fact table, Attributes table, Dimensions, Foreign Key	SQL-DDL, DML, UML Data Model Profile, UML Profile for Modeling Data Warehouse Usage	Enterprise Architect, Visual Paradigm, Oracle SQL Data Modeler	Microsoft SQL Server, Oracle, MySQL, PostgreSQL, IBM DB2
Big Data	Key-Value	Logical, Physical	Key, Value	SQL-DDL, Dynamo Query Language		Dynamo, Voldemort
	Document	Logical, Physical	Document, Primary Key	SQL-DDL, Javascript		MongoDB, CounchDB
	Wide-Column	Logical, Physical	Keyspace, Table, Column, Column Family, Super Column, Primary Key, Index	CQL, Groovy		Cassandra, HBase
	Graph	Logical, Physical	Node, Edge, Property	Cypher Query Language, SPARQL		Neo4j, AllegroGraph

(4) the concrete languages used to produce the data models and that apply the previous concepts; (5) the modeling tools that allow specifying diagrams using those languages and; (6) the database tools that support the data model. Table 2 presents the values of each feature for each approach. It is possible to verify that the majority of the data models are at a logical and physical level, with the exception of the ER model and the OLAP cube model, which are more abstract and defined at conceptual and logical levels. It is also possible to verify that Big Data has more data models than the other approaches, what can explain the work and proposals that have been conducted over the last years, as well as the absence of a *de facto* data model. In terms of concepts, again Big Data-related data models have a more variety of concepts than the other approaches, ranging from key-value pairs or documents to nodes and edges. Concerning concrete languages, it is concluded that every data model presented in this survey is supported by a SQL-DDL-like language. However, we found that only the operational databases and DWs have concrete languages to express their data models in a graphical way, like Chen's notation for ER model, UML Data Profile for Relational model or CWM [36] for multidimensional DW models. Also, related to that point, there are none modeling tools to express Big Data models. Thus, defining such a modeling language and respective supporting tool for Big Data models constitute an interesting research direction that fills this lack. At last, all approaches have database tools that support the development based on their

data models, with the exception of the ER model that is not directly used by DBMSs.

On the other hand, in terms of the ***Data Analytics Perspective***, Table 3 considers six features of analysis: (1) the class of application domains, which characterizes the approach suitability; (2) the common operations used in the approach, which can be reads and/or writes; (3) the operations types most typically used in the approach; (4) the concrete languages used to specify those operations; (5) the abstraction level of these concrete languages (Conceptual, Logical and Physical); and (6) the technology support of these languages and operations.

Table 3 shows that Big Data is used in more classes of application domains than the operational databases and DWs, which are used for OLTP and OLAP domains, respectively. It is also possible to observe that operational databases are commonly used for reads and writes of small operations (using transactions), because they need to handle fresh and critical data in a daily basis. On the other hand, DWs are mostly suited for read operations, since they perform analysis and data mining mostly with historical data. Big Data performs both reads and writes, but in a different way and at a different scale from the other approaches. Big Data applications are built to perform a huge amount of reads, and if a huge amount of writes is needed, like for OLTP, they sacrifice consistency (using "eventually consistency") in order to achieve great availability and horizontal scalability. Operational databases support their data manipulation operations (e.g. select, insert or delete) using SQL-ML, which has slight variations according to the technology used. DWs also use SQL-ML through the select statement, because their operations (e.g. slice, dice or drill down/up) are mostly reads. DWs also use SQL-based languages, like MDX and XMLA (XML for Analysis) [37], for specifying their operations. On the other hand, regarding Big Data technologies, there is a great variety of languages to manipulate data according to the different class application domains. All of these languages provide equivalent operations to the ones offered by SQL-ML and add new constructs for supporting both ETL, data stream processing (e.g. create stream, window) [34] and MapReduce operations. It is important to note that concrete languages used in the different approaches reside at logical and physical levels, because they are directly used by the supporting software tools.

5. Related Work

As mentioned in Section 1, the main goal of this paper is to present and discuss the concepts surrounding data

Table 3. Comparison of the approaches from the Data Analytics perspective.

Approaches / Features	Class of Application Domains	Common Operations	Operations	Concrete Languages	Abstraction Level	Technology Support
Operational	OLTP	Read/Write	Select, Insert, Update, Delete, Join, OrderBy, GroupBy	SQL-DML	Logical, Physical	Microsoft SQL Server, Oracle, MySQL, PostgreSQL, IBM DB2
Decision Support	OLAP	Read	Slice, Dice, Drill down, Drill up, Pivot	SQL-DML, MDX, XMLA	Logical, Physical	Microsoft SQL Server, Oracle, MySQL, PostgreSQL, IBM DB2, Microsoft OLAP Provider, Microsoft Analysis Services
Big Data	Batch-oriented processing	Read/Write	Map-Reduce, Select, Insert, Update, Delete, Load, Import, Export, OrderBy, GroupBy	Hive QL, Pig Latin	Logical, Physical	Hadoop, Hive Pig
	Stream processing	Read/Write	Aggregate, Partition, Merge, Join,	SQL stream	Logical, Physical	Storm, S4, Spark
	OLTP	Read/Write	Select, Insert, Update, Delete, Batch, Get, OrderBy, GroupBy	CQL, Java, JavaScript	Logical, Physical	Cassandra, HBase
	Interactive ad-hoc queries and analysis	Read	Select, Insert, Update, Delete, OrderBy, GroupBy	SQL-DML	Logical, Physical	Drill

modeling and data analytics, and their evolution for three representative approaches: operational databases, decision support databases and Big Data technologies. In our survey we have researched related works that also explore and compare these approaches from the data modeling or data analytics point of view.

J.H. ter Bekke provides a comparative study between the Relational, Semantic, ER and Binary data models based on an examination session results [38]. In that session participants had to create a model of a case study, similar to the Academic Management System used in this paper. The purpose was to discover relationships between the modeling approach in use and the resulting quality. Therefore, this study just addresses the data modeling topic, and more specifically only considers data models associated to the database design process.

Several works focus on highlighting the differences between operational databases and data warehouses. For example, R. Hou provides an analysis between operational databases and data warehouses distinguishing them according to their related theory and technologies, and also establishing common points where combining both systems can bring benefits [39]. C. Thomsen and T.B. Pedersen compare open source ETL tools, OLAP clients and servers, and DBMSs, in order to build a Business Intelligence (BI) solution [40].

P. Vassiliadis and T. Sellis conducted a survey that focuses only on OLAP databases and compare various proposals for the logical models behind them. They group the various proposals in just two categories: commercial tools and academic efforts, which in turn are subcategorized in relational model extensions and cube-oriented approaches [41]. However, unlike our survey they do not cover the subject of Big Data technologies.

Several papers discuss the state of the art of the types of data stores, technologies and data analytics used in Big Data scenarios [29] [30] [33] [42], however they do not compare them with other approaches. Recently, P. Chandarana and M. Vijayalakshmi focus on Big Data analytics frameworks and provide a comparative study according to their suitability [35].

Summarizing, none of the following mentioned work provides such a broad analysis like we did in this paper, namely, as far as we know, we did not find any paper that compares simultaneously operational databases, decision support databases and Big Data technologies. Instead, they focused on describing more thoroughly one or two of these approaches

6. Conclusions

In recent years, the term Big Data has appeared to classify the huge datasets that are continuously being produced from various sources and that are represented in a variety of structures. Handling this kind of data represents new challenges, because the traditional RDBMSs and DWs reveal serious limitations in terms of performance and scalability when dealing with such a volume and variety of data. Therefore, it is needed to reinvent the ways in which data is represented and analyzed, in order to be able to extract value from it.

This paper presents a survey focused on both these two perspectives: data modeling and data analytics, which are reviewed in terms of the three representative approaches nowadays: operational databases, decision support databases and Big Data technologies. First, concerning data modeling, this paper discusses the most common data models, namely: relational model and ER model for operational databases; star schema model and OLAP cube model for decision support databases; and key-value store, document-oriented database, wide-column store and graph database for Big Data-based technologies. Second, regarding data analytics, this paper discusses the common operations used for each approach. Namely, it observes that operational databases are more suitable for OLTP applications, decision support databases are more suited for OLAP applications, and Big Data technologies are more appropriate for scenarios like batch-oriented processing, stream processing, OLTP and interactive ad-hoc queries and analysis.

Third, it compares these approaches in terms of the two perspectives and based on some features of analysis. From the data modeling perspective, there are considered features like the data model, its abstraction level, its concepts, the concrete languages used to described, as well as the modeling and database tools that support it. On the other hand, from the data analytics perspective, there are taken into account features like the class of application domains, the most common operations and the concrete languages used to specify those operations. From this analysis, it is possible to verify that there are several data models for Big Data, but none of them is represented by any modeling language, neither supported by a respective modeling tool. This issue constitutes an open research area that can improve the development process of Big Data targeted applications, namely applying a Model-Driven Engineering approach [12]-[14]. Finally, this paper also presents some related work on the data modeling and data analytics areas.

As future work, we consider that this survey may be extended to capture additional aspects and comparison features that are not included in our analysis. It will be also interesting to survey concrete scenarios where Big Data technologies prove to be an asset [43]. Furthermore, this survey constitutes a starting point for our ongoing research goals in the context of the Data Storm and MDD Lingo initiatives. Specifically, we intend to extend existing domain-specific modeling languages, like XIS [44] and XIS-Mobile [45] [46], and their MDE-based framework to support both the data modeling and data analytics of data-intensive applications, such as those researched in the scope of the Data Storm initiative [47]-[50].

Acknowledgements

This work was partially supported by national funds through FCT—Fundação para a Ciência e a Tecnologia, under the projects POSC/EIA/57642/2004, CMUP-EPB/TIC/0053/2013, UID/CEC/50021/2013 and Data Storm Research Line of Excellency funding (EXCL/EEI-ESS/0257/2012).

References

[1] Mayer-Schönberger, V. and Cukier, K. (2014) Big Data: A Revolution That Will Transform How We Live, Work, and Think. Houghton Mifflin Harcourt, New York.

[2] Noyes, D. (2015) The Top 20 Valuable Facebook Statistics. https://zephoria.com/top-15-valuable-facebook-statistics

[3] Shvachko, K., Hairong Kuang, K., Radia, S. and Chansler, R. (2010) The Hadoop Distributed File System. 26*th Symposium on Mass Storage Systems and Technologies* (*MSST*), Incline Village, 3-7 May 2010, 1-10. http://dx.doi.org/10.1109/msst.2010.5496972

[4] White, T. (2012) Hadoop: The Definitive Guide. 3rd Edition, O'Reilly Media, Inc., Sebastopol.

[5] Dean, J. and Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communications*, **51**, 107-113. http://dx.doi.org/10.1145/1327452.1327492

[6] Hurwitz, J., Nugent, A., Halper, F. and Kaufman, M. (2013) Big Data for Dummies. John Wiley & Sons, Hoboken.

[7] Beyer, M.A. and Laney, D. (2012) The Importance of "Big Data": A Definition. Gartner. https://www.gartner.com/doc/2057415

[8] Duncan, A.D. (2014) Focus on the "Three Vs" of Big Data Analytics: Variability, Veracity and Value. Gartner. https://www.gartner.com/doc/2921417/focus-vs-big-data-analytics

[9] Agrawal, D., Das, S. and El Abbadi, A. (2011) Big Data and Cloud Computing: Current State and Future Opportunities. *Proceedings of the* 14*th International Conference on Extending Database Technology*, Uppsala, 21-24 March, 530-533. http://dx.doi.org/10.1145/1951365.1951432

[10] McAfee, A. and Brynjolfsson, E. (2012) Big Data: The Management Revolution. Harvard Business Review.

[11] DataStorm Project Website. http://dmir.inesc-id.pt/project/DataStorm.

[12] Stahl, T., Voelter, M. and Czarnecki, K. (2006) Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Inc., New York.

[13] Schmidt, D.C. (2006) Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, **39**, 25-31. http://dx.doi.org/10.1109/MC.2006.58

[14] Silva, A.R. (2015) Model-Driven Engineering: A Survey Supported by the Unified Conceptual Model. *Computer Languages*, *Systems & Structures*, **43**, 139-155.

[15] Ramakrishnan, R. and Gehrke, J. (2012) Database Management Systems. 3rd Edition, McGraw-Hill, Inc., New York.

[16] Connolly, T.M. and Begg, C.E. (2005) Database Systems: A Practical Approach to Design, Implementation, and Management. 4th Edition, Pearson Education, Harlow.

[17] Codd, E.F. (1970) A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, **13**, 377-387. http://dx.doi.org/10.1145/362384.362685

[18] Bachman, C.W. (1969) Data Structure Diagrams. *ACM SIGMIS Database*, **1**, 4-10. http://dx.doi.org/10.1145/1017466.1017467

[19] Chamberlin, D.D. and Boyce, R.F. (1974) SEQUEL: A Structured English Query Language. In: *Proceedings of the* 1974 *ACM SIGFIDET* (*Now SIGMOD*) *Workshop on Data Description*, *Access and Control* (*SIGFIDET*' 74), ACM Press, Ann Harbor, 249-264.

[20] Chen, P.P.S. (1976) The Entity-Relationship Model—Toward a Unified View of Data. *ACM Transactions on Database Systems*, **1**, 9-36. http://dx.doi.org/10.1145/320434.320440

[21] Tanaka, A.K., Navathe, S.B., Chakravarthy, S. and Karlapalem, K. (1991) ER-R, an Enhanced ER Model with Situation-Action Rules to Capture Application Semantics. *Proceedings of the* 10*th International Conference on Entity-Relationship Approach*, San Mateo, 23-25 October 1991, 59-75.

[22] Merson, P. (2009) Data Model as an Architectural View. Technical Note CMU/SEI-2009-TN-024, Software Engineering Institute, Carnegie Mellon.

[23] Kimball, R. and Ross, M. (2013) The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 3rd Edition, John Wiley & Sons, Inc., Indianapolis.

[24] Zhang, D., Zhai, C., Han, J., Srivastava, A. and Oza, N. (2009) Topic Modeling for OLAP on Multidimensional Text Databases: Topic Cube and Its Applications. *Statistical Analysis and Data Mininig*, **2**, 378-395. http://dx.doi.org/10.1002/sam.10059

[25] Gray, J., *et al*. (1997) Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, **1**, 29-53. http://dx.doi.org/10.1023/A:1009726021843

[26] Cattell, R. (2011) Scalable SQL and NoSQL Data Stores. *ACM SIGMOD Record*, **39**, 12-27. http://dx.doi.org/10.1145/1978915.1978919

[27] Gilbert, S. and Lynch, N. (2002) Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News*, **33**, 51-59.

[28] Vogels, W. (2009) Eventually Consistent. *Communications of the ACM*, **52**, 40-44. http://dx.doi.org/10.1145/1435417.1435432

[29] Grolinger, K., Higashino, W.A., Tiwari, A. and Capretz, M.A.M. (2013) Data Management in Cloud Environments: NoSQL and NewSQL Data Stores. *Journal of Cloud Computing*: *Advances*, *Systems and Applications*, **2**, 22. http://dx.doi.org/10.1186/2192-113x-2-22

[30] Moniruzzaman, A.B.M. and Hossain, S.A. (2013) NoSQL Database: New Era of Databases for Big data Analytics-Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, **6**, 1-14.

[31] Chang, F., *et al*. (2006) Bigtable: A Distributed Storage System for Structured Data. *Proceedings of the* 7*th Symposium on Operating Systems Design and Implementation* (*OSDI*' 06), Seattle, 6-8 November 2006, 205-218.

[32] Spofford, G., Harinath, S., Webb, C. and Civardi, F. (2005) MDX Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase. John Wiley & Sons, Inc., Indianapolis.

[33] Hu, H., Wen, Y., Chua, T.S. and Li, X. (2014) Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*, **2**, 652-687. http://dx.doi.org/10.1109/ACCESS.2014.2332453

[34] Golab, L. and Özsu, M.T. (2003) Issues in Data Stream Management. *ACM SIGMOD Record*, **32**, 5-14. http://dx.doi.org/10.1145/776985.776986

[35] Chandarana, P. and Vijayalakshmi, M. (2014) Big Data Analytics Frameworks. *Proceedings of the International Conference on Circuits*, *Systems*, *Communication and Information Technology Applications* (*CSCITA*), Mumbai, 4-5 April 2014, 430-434. http://dx.doi.org/10.1109/cscita.2014.6839299

[36] Poole, J., Chang, D., Tolbert, D. and Mellor, D. (2002) Common Warehouse Metamodel. John Wiley & Sons, Inc., New York.

[37] XML for Analysis (XMLA) Specification. https://msdn.microsoft.com/en-us/library/ms977626.aspx.

[38] ter Bekke, J.H. (1997) Comparative Study of Four Data Modeling Approaches. *Proceedings of the* 2*nd EMMSAD Workshop*, Barcelona, 16-17 June 1997, 1-12.

[39] Hou, R. (2011) Analysis and Research on the Difference between Data Warehouse and Database. *Proceedings of the International Conference on Computer Science and Network Technology* (*ICCSNT*), Harbin, 24-26 December 2011, 2636-2639.

[40] Thomsen, C. and Pedersen, T.B. (2005) A Survey of Open Source Tools for Business Intelligence. *Proceedings of the* 7*th International Conference on Data Warehousing and Knowledge Discovery* (*DaWaK*'05), Copenhagen, 22-26 August 2005, 74-84. http://dx.doi.org/10.1007/11546849_8

[41] Vassiliadis, P. and Sellis, T. (1999) A Survey of Logical Models for OLAP Databases. *ACM SIGMOD Record*, **28**, 64-69. http://dx.doi.org/10.1145/344816.344869

[42] Chen, M., Mao, S. and Liu, Y. (2014) Big Data: A Survey. *Mobile Networks and Applications*, **19**, 171-209. http://dx.doi.org/10.1007/978-3-319-06245-7

[43] Chen, H., Hsinchun, R., Chiang, R.H.L. and Storey, V.C. (2012) Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, **36**, 1165-1188.

[44] Silva, A.R., Saraiva, J., Silva, R. and Martins, C. (2007) XIS-UML Profile for Extreme Modeling Interactive Systems. *Proceedings of the* 4*th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*

(*MOMPES*'07), Braga, 31-31 March 2007, 55-66. http://dx.doi.org/10.1109/MOMPES.2007.19

[45] Ribeiro, A. and Silva, A.R. (2014) XIS-Mobile: A DSL for Mobile Applications. *Proceedings of the* 29*th Symposium on Applied Computing* (*SAC* 2014), Gyeongju, 24-28 March 2014, 1316-1323. http://dx.doi.org/10.1145/2554850.2554926

[46] Ribeiro, A. and Silva, A.R. (2014) Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development. *Journal of Software Engineering and Applications*, **7**, 906-919. http://dx.doi.org/10.4236/jsea.2014.711081

[47] Silva, M.J., Rijo, P. and Francisco, A. (2014). Evaluating the Impact of Anonymization on Large Interaction Network Datasets. In: *Proceedings of the* 1*st International Workshop on Privacy and Security of Big Data*, ACM Press, New York, 3-10. http://dx.doi.org/10.1145/2663715.2669610

[48] Anjos, D., Carreira, P. and Francisco, A.P. (2014) Real-Time Integration of Building Energy Data. *Proceedings of the IEEE International Congress on Big Data*, Anchorage, 27 June-2 July 2014, 250-257. http://dx.doi.org/10.1109/BigData.Congress.2014.44

[49] Machado, C.M., Rebholz-Schuhmann, D., Freitas, A.T. and Couto, F.M. (2015) The Semantic Web in Translational Medicine: Current Applications and Future Directions. *Briefings in Bioinformatics*, **16**, 89-103. http://dx.doi.org/10.1093/bib/bbt079

[50] Henriques, R. and Madeira, S.C. (2015) Towards Robust Performance Guarantees for Models Learned from High-Dimensional Data. In: Hassanien, A.E., Azar, A.T., Snasael, V., Kacprzyk, J. and Abawajy, J.H., Eds., *Big Data in Complex Systems*, Springer, Berlin, 71-104. http://dx.doi.org/10.1007/978-3-319-11056-1_3