Data & Knowledge Engineering 108 (2017) 50–67

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Big data technologies and Management: What conceptual modeling can do

Veda C. Storey[a,*], Il-Yeol Song[b]

[a] J. Mack Robinson College of Business, Dept. of Computer Information Systems, Georgia State University, Atlanta, Georgia
[b] College of Computing & Informatics, Drexel University, Philadelphia, PA 19104, USA

A B S T R A C T

The era of big data has resulted in the development and applications of technologies and methods aimed at effectively using massive amounts of data to support decision-making and knowledge discovery activities. In this paper, the five Vs of big data, volume, velocity, variety, veracity, and value, are reviewed, as well as new technologies, including NoSQL databases that have emerged to accommodate the needs of big data initiatives. The role of conceptual modeling for big data is then analyzed and suggestions made for effective conceptual modeling efforts with respect to big data.

1. Introduction

Big data is widely recognized as referring to the very large amounts of data, both structured and unstructured, that organizations are now capable of capturing and attempting to analyze in a meaningful way so that data-driven decision analysis and actionable insights can be obtained. Doing so has required the development of techniques and methods for analysis, new ways to structure data, and interesting applications in science and in management (e.g., [14,5,1]). Although the value of big data has sometimes been challenged, the big data landscape continues to grow [22].

The objective of this paper is to examine the progression of big data in an effort to: identify the challenges that exist; and specify the role that conceptual modeling can play in advancing work in this important area. The next section defines and describes big data and its recognized, inherent characteristics. Then, new and emerging big data technologies are presented before analyzing the specific role that conceptual modeling can play in understanding and advancing research and applications of big data.

2. Big Data

The volume of data has grown exponentially over the past decade, to the point where the management of the data asset by traditional means is no longer possible [26]. As shown in Fig. 1, big data trends have been enabled by advances in computing technologies, which facilitated the sudden explosion of data from various sources such as the Web, social media, and sensors. The flood of data brought about the emergence of a data-driven paradigm to take advantage of the newly available computing technologies. Big data technologies materialized the data-driven paradigm, making it increasingly sophisticated and useful.

Big data refers to the high volume, velocity, and variety of information assets that demand new, innovative forms of processing for enhanced decision making, business insights, and process optimization [18]. As a relatively new concept, the basic notion of big data includes the techniques and technologies required to manage very large quantities of data. In addition to the technologies,

* Corresponding author.
E-mail addresses: VStorey@gsu.edu (V.C. Storey), song@drexel.edu (I.-Y. Song).

http://dx.doi.org/10.1016/j.datak.2017.01.001

Available online 27 February 2017

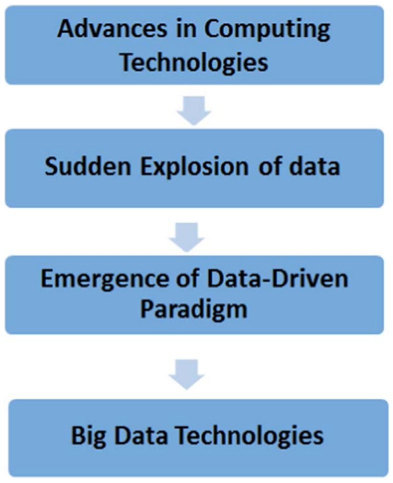

Advances in Computing Technologies
Sudden Explosion of data
Emergence of Data-Driven Paradigm
Big Data Technologies

Fig. 1. Factors leading to Big Data.

skilled professionals are needed with analysis and design skills to appropriately manage this resource [2,16].

Mayer-Schonberger and Cukier [21] argue that big data will change the way people live, work, and think, although it requires that many obstacles be overcome. The data must be obtained, processed, and effectively used, raising related issues on how big data will be represented and modelled. Understanding the challenges associated with big data representation and modeling, though, first requires an understanding of the characteristics of big data.

2.1. The Vs of big data

Big data, as traditionally characterized by the "3Vs" of volume, variety, and velocity, have emerged from advances in sensing, measuring, and social computing technologies (Gartner.com). In addition to these Vs, veracity (accuracy) and, especially, *value*, are important. Each of the Vs has its own unique challenges. The volume is too big, the variety requires both structured and unstructured analysis, and the velocity is so fast that we might not even have time to identify reasonable questions to ask [8]. The veracity leads to uncertainty, and the volume competes with velocity [27]. It is the value, however, that is the most time-consuming to extract, and difficult to ascertain. Fig. 2 summarizes the "5 V" challenges dominant in big data practice and research efforts.

Volume: The large volume of data has resulted in data availability coming from diverse, often location-dependent, data streams containing various kinds of data that are being generated at a very high velocity from huge banks of physical, digital, and human sensors [10]. The data sources include wearable technologies, cloud-based service (e.g., Amazon web services), enterprise data warehouses (EDW), and NoSQL databases [40]. The scale is now terabytes, petabytes, and exabytes. The volume challenge is being

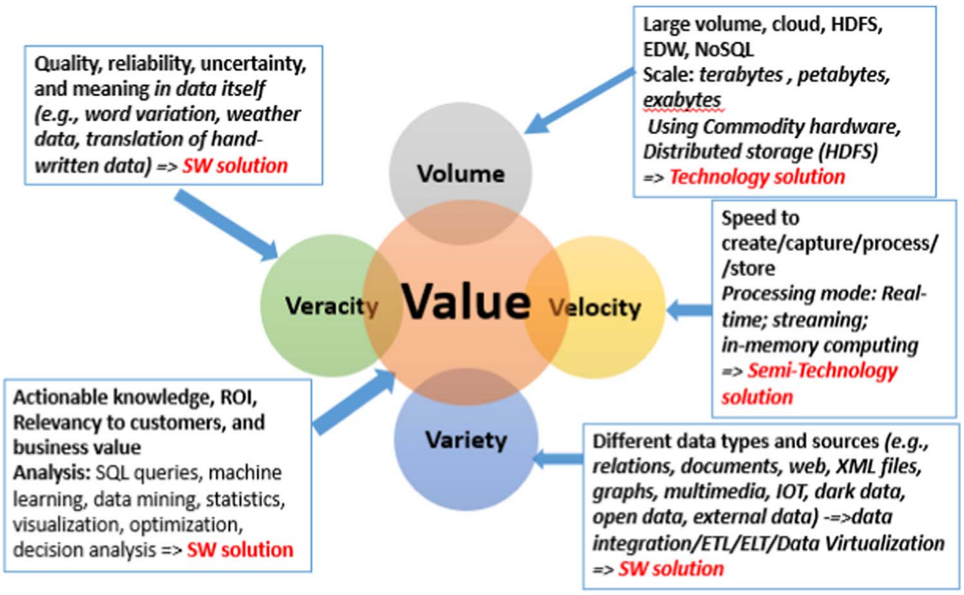

Quality, reliability, uncertainty, and meaning in data itself (e.g., word variation, weather data, translation of hand-written data) => SW solution
Large volume, cloud, HDFS, EDW, NoSQL
Scale: terabytes , petabytes, exabytes
Using Commodity hardware, Distributed storage (HDFS)
=> Technology solution
Volume
Veracity
Value
Velocity
Speed to create/capture/process/ /store
Processing mode: Real-time; streaming; in-memory computing
=> Semi-Technology solution
Actionable knowledge, ROI, Relevancy to customers, and business value
Analysis: SQL queries, machine learning, data mining, statistics, visualization, optimization, decision analysis => SW solution
Variety
Different data types and sources (e.g., relations, documents, web, XML files, graphs, multimedia, IOT, dark data, open data, external data) -=>data integration/ETL/ELT/Data Virtualization
=> SW solution

Fig. 2. The 5 Vs of Big Data.

addressed, technologically, by using commodity hardware and the Hadoop Distributed File System (HDFS).

Velocity: The velocity is the speed to create, capture, extract, process, and store data. A semi-technology solution is needed to deal with the velocity challenge, with the software solution portion having real-time processing, streaming and in-memory computing.

Variety: Different data types and sources provide relations (from relational databases), documents, Web data, XML files, sensor data, multimedia files, and so forth. The variety challenge is primarily addressed by software solutions because the integration of heterogeneous data requires an extensive software effort to handle the variety.

Veracity: The veracity refers to the imprecision of the data. It raises issues of quality, reliability, uncertainty, incompleteness, and the meaning in the data itself (e.g., word variation, weather data, translation of hand-written data). Eventually, though, the veracity must be consistent in order to be processed in an automated manner. The veracity challenge should be addressed by software solutions.

Value: The value of big data can be difficult to ascertain. Reasonable measures focus on: identifying "actionable" knowledge; computing, if possible, a return on investment (ROI); identifying relevancy to customers (as evidenced by much research on text mining and sentiment analysis); and other measures. The needed analysis of big data to identify such value may occur in various ways including: traditional SQL-type queries, machine learning techniques, data mining, statistics, optimization, and decision support analysis. The results may be represented in different forms, including traditional, standard and ad hoc report generation, and visualization. The value challenge is most difficult to achieve as its software solutions must be addressed within the context of the business or problem domain. Furthermore, available and qualified data scientists who can make sense of big data with a proper understanding of the domain and who are comfortable using analytical tools are not easy to find.

The extension to 7 Vs incorporates validity and volatility [13]. Applicable, additional Vs include visualization, variability, vulnerability, visibility, vagueness and, no doubt, others, all of which must be managed. However, "understanding" and dealing with the semantics of the data remains a big challenge [31,32,36].

2.2. Infrastructure to manage big data

To accommodate big data initiatives, firms have been building an infrastructure to manage it, treating big data as a corporate asset. The infrastructure requires having the correct tools in place and, generally, consists of the following components [12]:

- *Data stack*: consists of structured, as well as unstructured, data.
- *Big data ecosystem*: consists of search and visualization, data orchestration, and data access. Professional services for big data include the hardware, such as the computer, storage and networks. Specific database-related services are SQL-based systems, NoSQL systems, and Hadoop and its ecosystems.
- *Enterprise information management*: focuses on issues related to data governance, data integration, data quality, data visualization, and master data management.
- *Data Science platforms and tools*: have a variety of tools for pattern extraction and visualization of results. These include: machine learning algorithms; predictive analytics [7]; prescription techniques (e.g., simulation with alternative variables and subsets of data); descriptive techniques (e.g., statistics and assessment techniques for historical reporting); and reports (e.g., represented by scorecards or dashboards).

2.3. Data-driven paradigm

The data-driven paradigm [21] has implications for analysis and decision making. The important differences between traditional data management and this newer paradigm are summarized as follows.

Whole data set sampling. The entire set of data can be used in analysis, instead of sampling. Lin et al. [19] refer to this as sampling with a size of 'N' motivated, in part, by falling costs of data collection and analysis, and leading to collecting as much data as possible [15].

Accept the probably of prediction. As illustrated in the movie *Moneyball*, data analysis uses statistics to understand, for example, the sport of baseball, in a manner not previously considered. (http://blog.minitab.com/blog/the-statistics-game/moneyball-shows-the-power-of-statistics).

Big data outperforms experts. The computer system, DeepBlue [11], defeated a world chess champion based on techniques developed in artificial intelligence. Amazon reviews (Amazon.com), as contributed by web citizens, have illustrated their value over experts' recommendations.

Accept correlation. The identification of causality may not always be possible. Recommender systems, such as those used by Amazon, identify items that people are likely to purchase together. However, they do not identify the reason why this is so.

Datafication. This trend is to convert many unusual aspects of social life into computerized data that can be analyzed [21]. The key idea is to digitize the behaviors or thoughts of human, society, and machines to develop new applications.

Quantify/measure as many granular data as possible. Granular data refers to the level of detail in which the data is stored (e.g., low level of detail to higher summarized data). With big data technologies, the more granular data has greater potential for extracting business value and increasing the possibility of being reused in multiple applications.

Data speaking. Let the data speak for itself. This reflects the movement of value from physical items to brands, ideas, and intellectual rights.

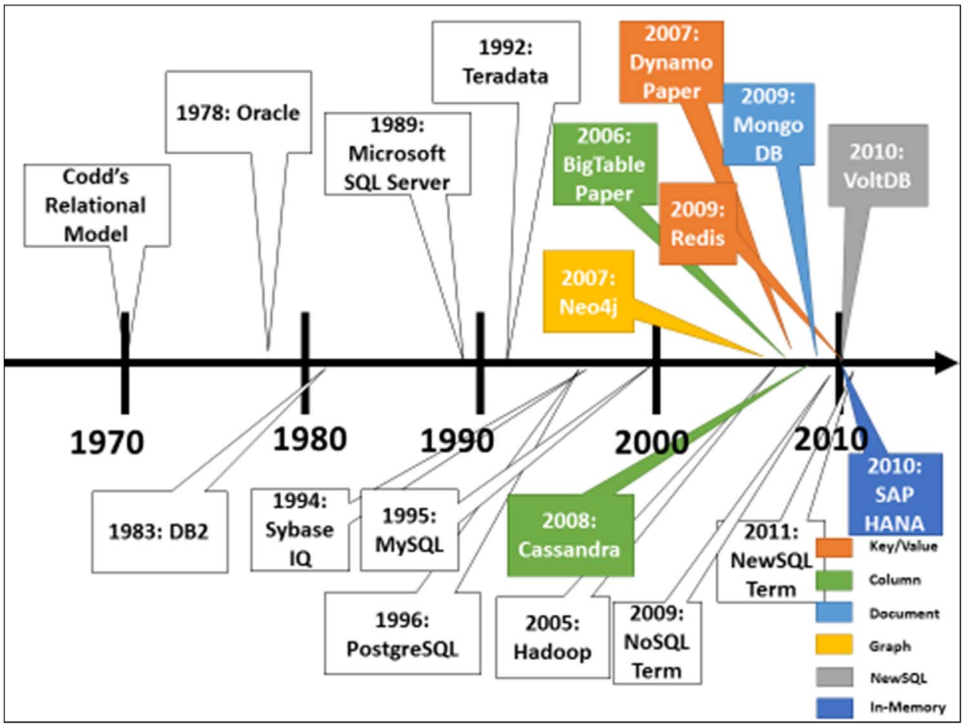


Fig. 3. Data Management Evolution.

2.4. Growth and Challenges

Advances in storage, networks, CPU, and bandwidth are remarkable. The storage costs ($ per terabyte), for example, have decreased from $14,000,000 (1980) to approximately $50. The number of nodes a company might have has risen from 1(1969) to 1 billion hosts. The cost per GFLOPS is the cost for a set of hardware that would, theoretically, operate at one billion floating-point operations per second. The CPU ($ per GFLOPS) costs have plummeted from $1,100,000,000 (1961) to $0.08. Bandwidth ($per Mbps) that was approximately $1200 in 1998, now costs approximately $5.

The sudden explosion of data from automated capture, sensors, and other sources, which led to the data-driven paradigm, has been embraced for real-time decision making, as well as other, data-intensive, activities. The size of the world big data market (hardware, software, and services) has been estimated to grow to over $50 billion [44]. Big data is driven, fundamentally, by advances in computing technologies with faster, cheaper, and more powerful processing capabilities due to Moore's law and other, alternative ways to progress computing capabilities.

There are many challenges associated with big data [37], including the difficulty of humans to comprehend its scale. The volume is too big and the velocity too fast [8]. The variety and veracity of data are more challenging in developing software solutions. Value is most intriguing, and realized by the opportunities big data provides for innovative solutions to problems. Furthermore, the impact on society and business is substantial.

3. New database technologies for big data

Database technologies have emerged that are specifically designed and employed for big data. Fig. 3 shows the timeline during which milestones related to the development of big data techniques have occurred. Starting in the 1970s, the relational database management approach [6] was introduced. In the 1980s, it started to become a reliable and efficient structured approach to managing data. In a relational model, data is represented as tuples in tables, called relations. Efforts were made to understand how to translate conceptual models to relational models (e.g., [38,35]). As the volume of data being stored in a database increased, notions associated with "very large" databases emerged, as recognized by the *Very Large Database* conferences and journal. These very large databases (in the range of terabytes) required a different storage model due to: (1) the tremendous overhead in the relational model (although less of a consideration today); and (2) different retrieval and usage demands. The simple retrieval of a personal record of a customer, John Doe, for example, was not enough. Instead, a single result was desired for all those who viewed catalog listings for something similar to "men's socks."

By the early 1990s, a terabyte of data was considered to be a large amount of data, in response to the desire and need to capture more and more data. Consequently, different types of relational data management tools and systems continued to dominate the market (e.g., Oracle, IBM DB 2, Microsoft SQL Server). During the first decade of this century, systems were developed that were less reliant on the relational database structure, deviating to keys, columns, documents, graphs, in memory management, and other techniques. Hadoop, for example, is an open source framework for handling unstructured data using a parallel-processing technique called MapReduce, originally developed by Google. Further development included a series of NoSQL and NewSQL databases. Most of these new breeds of databases support scalability and performance, using the scale-out architecture.

3.1. What's wrong with traditional RDBMSs?

Traditional relational database management systems (RDMS) are simply not capable of handling big data. The data is too big, too fast, and too diverse to store and manipulate. Relational databases require a schema before writing to the database, which is too rigid to manipulate volumes of real-time data with diverse data structures. The ACID properties (atomicity, consistency, isolation, and durability) are too strict for some applications. The RDBMS clusters are SPOF (Single Point of Failure) and expensive, and have impedance mismatch (aggregate versus atomic data). This all led to requirements for new architectures and new transaction management such as BASE (**B**asically **A**vailable, **S**oft state, **E**ventual consistency), which relaxes the ACID properties in distributed data management systems. BASE is common in NoSQL systems.

3.2. Hadoop

Map/Reduce is a programming framework with automatic parallelization. The Map part applies to the input data. It emits reduction keys and values with the output sorted and partitioned for the Reduce aspect. The Reduce function is applied to data grouped by the reduction key. The reduce function "reduces" data in the sense that it can aggregate data by adding selected values. The Map and Reduce can be chained together for complex computations. The result is extreme *scalability,* well-suited for scale-out architecture that uses low cost commodity hardware with fault-tolerate features. Hadoop can process and store large amounts of structured, unstructured and semi-structured data.

Hadoop (hadoop.apache.org) is an open source version of the Map/Reduce algorithm, which was created to analyze large amounts of unstructured data and has become a de facto standard in big data analytics. In traditional database use, a query is written in a structured query language, the data is accessed as stored in a relational database, and the result obtained. These types of queries, however, can be limited, so the desired output may not be obtained. Using Hadoop, unstructured data can be combined in many ways to facilitate data mining for useful patterns. Hadoop 1.0 is a single-use system for batch applications; Hadoop 2.0 is a multi-use data platform supporting batch, interactive, streaming, and graph-data applications. The Hadoop ecosystem progresses from data storage, data processing, and data access, to data management as defined below.

Data storage – HDFS (Hadoop distributed file system) and HBase (column database storage).

Data processing – MapReduce (automatic parallel data processing).

Data access – Hive (SQL-like), Pig (data flow), Mahout (machine learning), Avro (data serialization and remote procedure protocol), Sqoop (relational database management connector).

Management – Oozie (workflow), Chukwa (monitoring), Flume (monitoring), and ZooKeeper (management).

Hadoop is inherently scalable and good for processing a large amount of data with automatic load balancing. Hadoop, however, is too dependent on HDFS when multiple iterations are needed and still requires significant manual coding to implement complex operations such as joins based on multipole fields. These limitations have brought a new memory-resident parallel processing framework, called Spark.

3.2.1. Apache spark

Apache Spark is an in-memory centric computing platform, designed specifically for large scale analytical processing (http://spark.apache.org/). It is a fast and general engine with a simple and expressive programming model for supporting a wide range of applications, including ETL (Extract, Transform, Load), machine learning, stream processing, and graph computation. Apache Spark is 100 times faster than Hadoop MapReduce in memory, or 10 times faster on disk. Spark has an advanced DAG (Directed Acyclic Graph) execution engine that supports cyclic data flow and in-memory computing [30]. Eighty high-level operators make it easy to build parallel applications, with interactive use from the Scala, Python and R shells. It combines SQL, streaming, and complex analytics. Spark has a stack of libraries that can be combined in a single application, and include SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming.

Spark can access diverse data sources such as HDFS (Hadoop Distributed File Sharing), Cassandra (column based database), HBase (Hadoop's database), Hive, and Tachyon. It uses Resilient Distributed Datasets (RDDs), which are fault-tolerant distributed memory abstractions that avoid replication. Spark can interactively query 1 to 2 terabytes of data in less than one second.

Whereas Hadoop is good for batch applications, Spark is good for running real-time or iterative applications such as machine learning or graph processing and is easier to program than Hadoop.

3.3. NOSQL and NewSQL databases

NoSQL databases are "Not Only SQL" to indicate that SQL can also be used even though they represent the data management systems invented to manage non-relational data. They are in the forms of key-value stores, column stores, document stores, and graph databases. NoSQL databases [25,29] have fewer consistency restrictions than conventional relational databases. They provide

Table 1
Comparison of SQL, NoSQL, and NewSQL Database.

	Old SQL	NoSQL	NewSQL
Relational	Yes	No	Yes
SQL	Yes	No	Yes
Column stores	No	Yes	Yes
Scale	Limited	Yes (Horizontally)	Yes (Horizontally)
Eventually consistent		Yes	
BASE (Basically Available, Soft state and Eventually consistent)	No	Yes	No
Handles large (big) volumes of data	No	Yes	Yes
Schema-less	No	Yes	No

specific data models for specific applications. These NoSQL systems use eventual consistency based on the CAP (consistency, availability, partition) theorem, which means that each system supports either high availability of the data or consistency when data are partitioned in a network.

NewSQL [34] is a class of new breed databases that have the strengths of both relational and NoSQL databases. Thus, they support SQL and the ACID properties, and are built on the scale-out architecture, supporting scalability and fault tolerance. Thus, NewSQL databases provide a scalable performance comparable to NoSQL systems for OLTP workloads. They have limited support for "variety" due to the need of a schema (e.g,: Google spanner, VoltDB, MemSQL, NuoDB, Clustrix). Table 1 compares standard SQL, NoSQL and NewSQL.

3.3.1. In-memory computing

In-memory computing has all data in RAM rather than on disk. so there is no disk buffer. In-memory databases (IMDBs) are ACID-compliant relational databases, offering SQL. Durability is supported by snapshots, transaction logging, and other features. For an In-Memory Database, the RDBMS stores data in memory with SQL support. The In-Memory Data Grid (IMDG) has several characteristics. It stores data in the RAM of servers distributed over a cluster. There is greater ability to scale-out than with IMDB, but less sophisticated SQL support. It has limited distributed SQL querying and indexing capabilities. The IMDG uses a key/value data structure that provides simple but fast retrieval as well as flexibility for application developers.

3.3.2. Hybrid cloud computing

Cloud computing, as defined by the National Institute of Standards and Technology, is: "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models" ([23], p.3). Its main characteristics are: on-demand self-service; broad network access; resource pooling; rapid elasticity; and measured services. Service models are *Software as a Service (SaaS), Platform as a Service (PaaS),* and *Infrastructure as a Service (IaaS).* The four deployment models are: Private, Community, Public, and Hybrid.

With respect to *Infrastructure as a Service*, the applications, data, runtime, middleware, and operating system are managed by the customer, with the vendor taking responsibility for networking, storage, servers, and virtualization. With *Platform as a Service*, the customer manages the applications and data only, whereas for *Software as a Service,* the vendor takes full responsibility for managing all of the above functions.

The hybrid cloud approach is widely adopted for managing big data. Hybrid cloud computing stores private, sensitive, and critical data in an on-premise server, and sharable data in a public cloud, making it valuable for dynamic or highly changeable workloads. This approach is shown in Fig. 4. The intended advantage of the hybrid cloud is to enable workloads to move between private and public clouds as organization's computing requirements change, thus providing businesses with greater flexibility and more data management options.

The *Integration Platform as a Service (iPaas)* is a suite of services. It connects cloud-based processes, services, applications and data between or across multiple organizations and is intended to support the development, execution, and governance of integration flows (gartner.com). It acts as a hybrid integration platform where different business units can connect to enterprise, and other services; customers and suppliers can be connected; and seamless integration with the Internet of Things, social media, and mobile devices, can all occur.

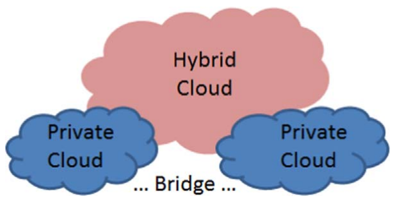


Fig. 4. Hybrid Cloud Combines Private and Public.

Cloud computing is a strategy that provides worry-free maintenance, rapid scalability, and incremental cost based on the usage for organizations. However, utilizing cloud computing requires good data governance, high quality metadata, and well-understood data integration processes.

3.3.3. Trend: ETL versus ELT

In the traditional ETL (Extract, Transform, and Load), the transformation is performed in the pre-processing stage before data is loaded into the server. With big data, where a large amount of data is rapidly generated, ELT (Extract, Load, and Transform), the transformation takes place in the target database after the data is loaded first. Data Virtualization enables data to stay in one place and not be transformed. Instead, data is transformed at run time when data is needed. Data virtualization provides a layer of software abstraction on the top of multiple data sources, without having to move the data from one place to another. This technology allows the creation of virtual data marts or real-time data integration platforms for analytics. The results impact data warehousing in that enterprise data warehouses can now incorporate data from: traditional relational databases, big data clusters, and real time stores. The data are used for a variety of applications including traditional OLAP (online analytic processing) and dashboards, as well as machine learning and parameterized ad hoc reporting.

3.4. Summary of big data technologies

Big data technologies are very complex and need to evolve in order to address issues associated with the Web, mobile, social media, cloud computing, and big data analytics. Hadoop ecosystems may evolve, but are still good for large sequential batch processing. Spark is becoming dominant and effective for interactive, real-time, parallel processing. NoSQL systems need a standard query language and discipline for modeling. In-memory computing is widely used in NoSQL and NewSQL, and is useful for analytics. Hadoop and enterprise data warehouses co-exist, with enterprise data warehouses evolving into an integrated data management platform. The use of cloud computing has increased, but hybrid cloud integration and security issues remain as important issues to be addressed. Big data continues to be the driving force for many applications, including the Internet of Things and other applications such as smart health and smart cities.

4. NoSQL databases

NoSQL refers to open-sourced, distributed, and non-relational databases. Horizontal-scaling enables scalability and fault tolerance. It is schemaless, allowing new data type to be dynamically added to the database, increasing write-performance. Most NoSQL systems adopt BASE (Basically **A**vailable, **S**oft state, **E**ventual consistency), as opposed to ACID (properties, atomicity, consistency, isolation, and durability), for transaction management to increase availability and performance rather than strict consistency.

NoSQL was specifically designed to address the needs of big data, big users, and cloud computing. It supports unstructured or non-relational data types (nested structure, column families, document, JSON (JavaScript Object Notation), BSON (binary serialization of JSON), and graph). NoSQL exhibits several other important characteristics. It is schema-less (schema-on-read, implicit schema). It can scale out massively at low cost and with fast retrieval (elastic scaling), providing low cost operational management for a large number of users. It supports scalability, high performance, and fault-tolerance, and is designed for real-time, non-uniform big data.

Activities important for dealing with big data issues include: scalability, schema flexibility, ease of development, cost, and availability of deployment options [9]. The shift from relational databases to NoSQL Databases is spurred as well by the need for flexibility both in the scaling model and the data model. In terms of scaling, in relational databases, scaling up is accomplished by adding a bigger server when additional capacity is needed. In NoSQL, scale out means that, instead of acquiring a bigger server, one can add more commodity servers.

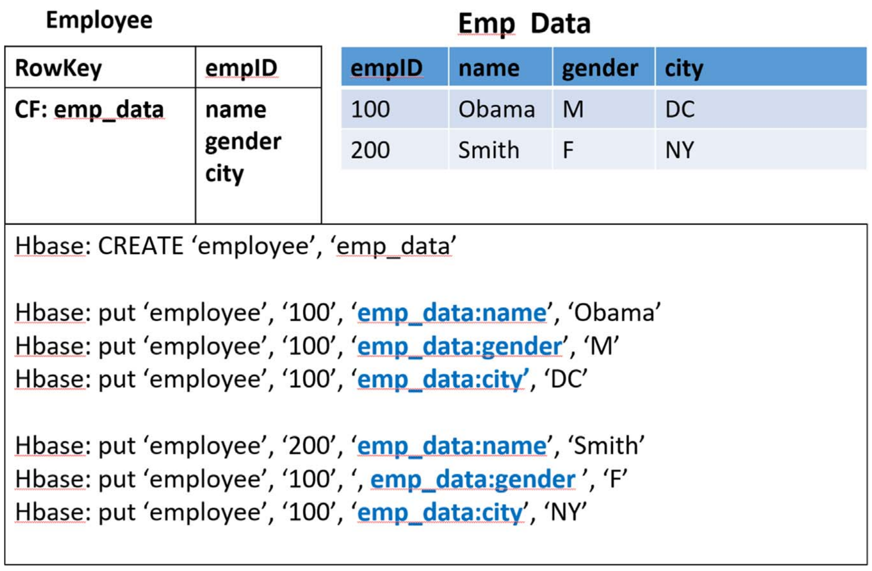
4.1. NoSQL database modeling

From a data modeling perspective, NoSQL is schemaless, has no join operation, and handles redundancy by embedding and linking. These characteristics facilitate the ease of application development by simplifying the mapping between the memory structure and the database structure. A relational database has a "schema-on-write," whereas NoSQL has "schema-on-read." NoSQL has a dynamic column family with column names defined at the time of data insertion, and an implicit schema, defined after coding. Fig. 5 shows an example of a dynamic column family in HBase, where the "rowkey" represents the identifier of employee data.

Dynamic Column Generation is shown in Fig. 6. Fig. 7 summarizes the comparison.

From a data modeling perspective, several major differences emerge. With respect to impedance mismatch, the relational database has a normalized atomic data. NoSQL has non-relational data to handle the *variety* of the 5Vs and a denormalized structure. The aggregate is the unit of operation and consistency. NoSQL does not support the join operation or foreign keys, as relational databases do. Rather, joins are performed in applications. As a result, most searching is performed against a single table, which must include all of the related data.

NoSQL databases manage redundancy control by embedding and linking. Embedding involves denormalizing 1:N and M:N relationships and collapsing subclass entities into superclass entities. It is efficient for retrieval and convenient for consistency control. Linking (referencing) involves adding a key to the object and is good when the referenced objects are static and the


Employee

RowKey	empID
CF: emp_data	name gender city

Emp Data

empID	name	gender	city
100	Obama	M	DC
200	Smith	F	NY

```
Hbase: CREATE ‘employee’, ‘emp_data’

Hbase: put ‘employee’, ‘100’, ‘emp_data:name’, ‘Obama’
Hbase: put ‘employee’, ‘100’, ‘emp_data:gender’, ‘M’
Hbase: put ‘employee’, ‘100’, ‘emp_data:city’, ‘DC’

Hbase: put ‘employee’, ‘200’, ‘emp_data:name’, ‘Smith’
Hbase: put ‘employee’, ‘100’, ‘, emp_data:gender ’, ‘F’
Hbase: put ‘employee’, ‘100’, ‘emp_data:city’, ‘NY’
```

Fig. 5. Dynamic column family in HBase.

relationships do not change as found, for example, between publisher and book.

Most NoSQL databases lack full ACID compliance for guaranteeing transactional integrity and data consistency. Many NoSQL databases do not guarantee consistency, by design, because many applications need to handle potential inconsistencies. *Eventual consistency* limits the use of NoSQL databases for mission-critical transactional applications. There are many variations, however, even within the same category of NoSQL systems.

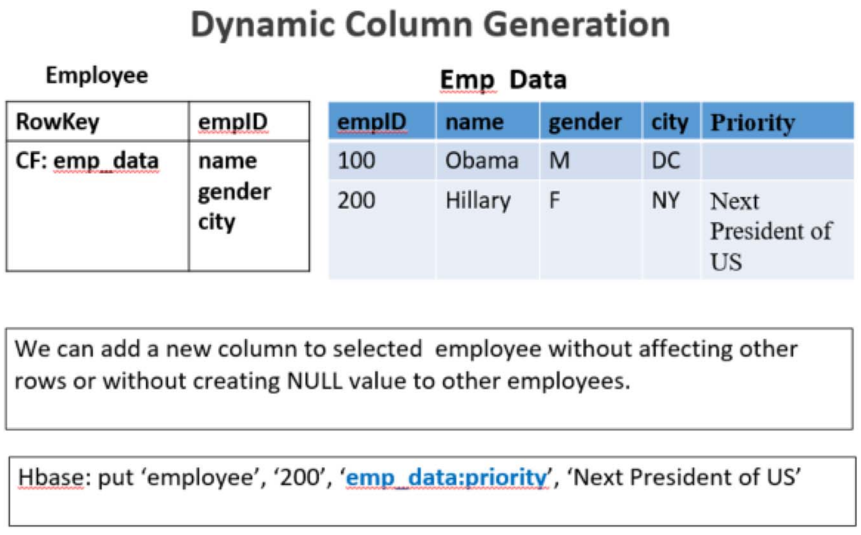
4.2. Types of NoSQL databases

There are four main types of NoSQL databases, each with different levels of scalability, flexibility, complexity, and functionality (academy.datastax.com).

4.2.1. Key-value Store

Key-Value Store databases store data in a schema-less way, with all of the data consisting of an indexed key and a value (e.g., Oracle NoSQL and Redis [28]). A simple hash table can only be accessed by Key to retrieve Value. Value can be any type of data (“aggregates”) in any size, and treated as a blob. Some models provide indexes on values. Designed to handle massive load, the goal is to provide a very fast access using the Key, much faster than using SQL.

Both Key and Value can be a complex compound object. This is good for data that is only accessed through a single key and where


Dynamic Column Generation

Employee

RowKey	empID
CF: emp_data	name gender city

Emp Data

empID	name	gender	city	Priority
100	Obama	M	DC	
200	Hillary	F	NY	Next President of US

We can add a new column to selected employee without affecting other rows or without creating NULL value to other employees.

```
Hbase: put ‘employee’, ‘200’, ‘emp_data:priority’, ‘Next President of US’
```

Fig. 6. Dynamic column generation.

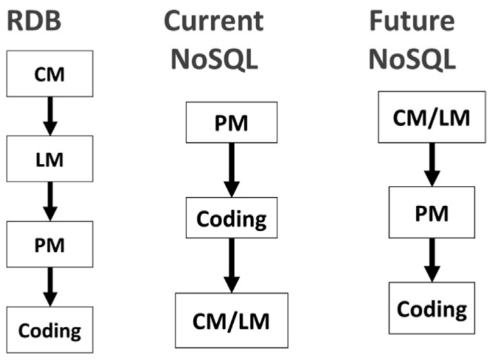


Fig. 7. Comparison from current relational databases to current NoSQL to future NoSQL (CM=Conceptual Model; LM=Logical Model; PM=Physical Model).

multiple records do not need to be related. To read the components of postings, we need to parse all of the objects. The advantages are that it has very fast random access via Key, is scalable, easy to distribute across clusters, and provides a simple model as a hash table. The disadvantages are that there is no complex filtering query, the join needs to be performed in the applications, and there is no mechanism for supporting multi-record consistency.

4.2.2. Column store

A Column Store (also known as wide-column stores) database was motivated by Google's BigTable. Instead of storing data in rows, data tables are stored as sections of columns of data. It is an extension of Key-Value Store database where columns can have a complex structure, rather than a blob value. A Column Store database supports complex modeling structures (nested tables, repeating groups, set, list, etc.). A column family (called Super Family in Cassandra [17]) is a group of related columns, guaranteed to maintain consistency. Column Store have high performance and a highly scalable architecture.

The key structure of a typical column store consists of {row_key, column family name, column qualifier name, timestamp}. A timestamp supports versioning; by default, the most recent value is retrieved. Thus, rapid temporal analysis of a column value is possible. A row key can be a composite key, but is stored as a single key.

Modeling Aspects of Column Store Databases: Column stores support rich modeling structures (columns, composite columns, denormalized structures, nested tables, aggregates, secondary indexes, map, set, and list). The data is stored so that all of the data can be accessed together. Denormalization is common, but it brings redundancy and consistency problems.

One should consider query refinement first, and then design the physical structure for query performance. Developers often

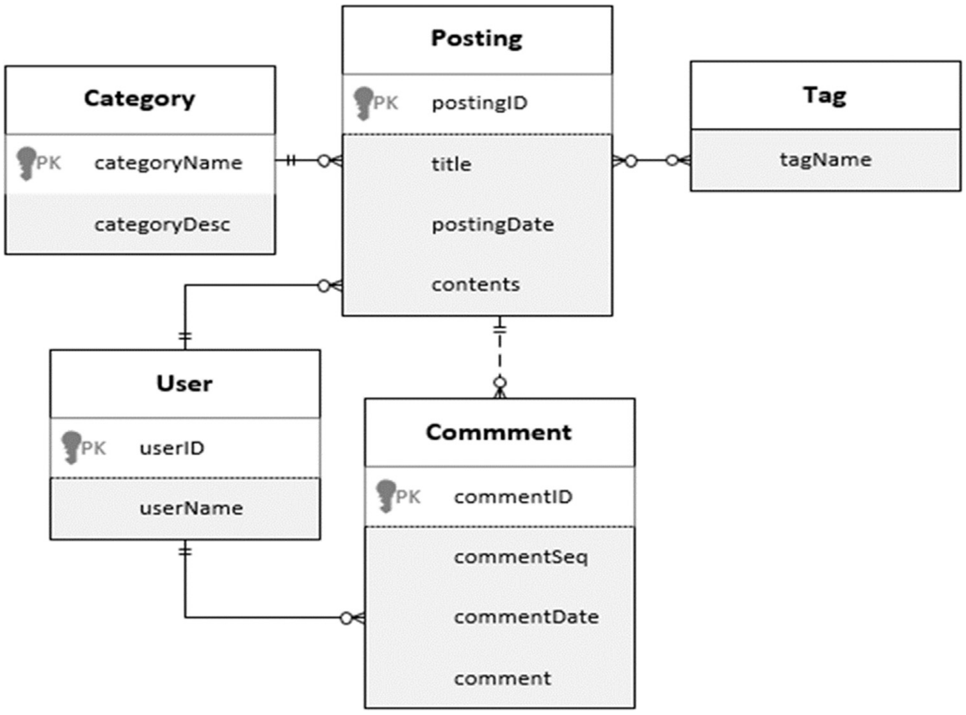
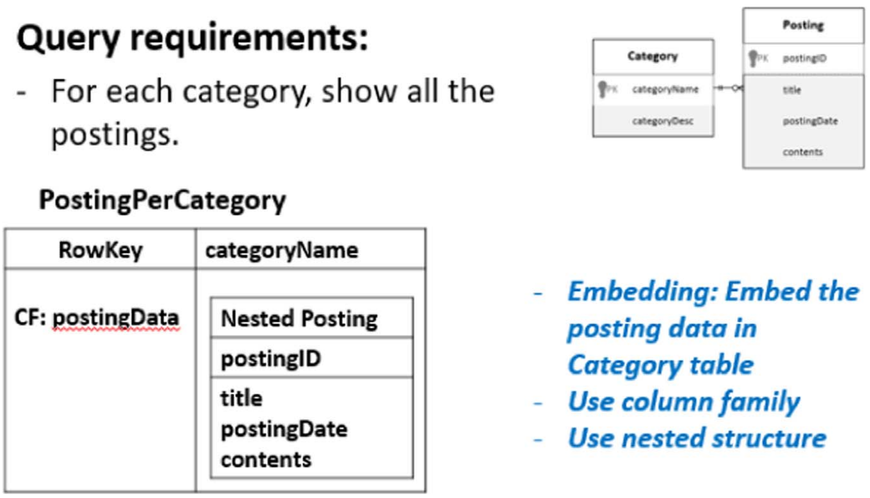


Fig. 8. Entity-relationship model for a NoSQL Column Store Database.


Query requirements:

- For each category, show all the postings.

PostingPerCategory

RowKey	categoryName
CF: postingData	Nested Posting postingID title postingDate contents

- *Embedding: Embed the posting data in Category table*
- *Use column family*
- *Use nested structure*

Fig. 9. 1:N Relationship with Embedding for Query 1.

create one aggregate table per query requirement. There are multiple ways in which to model a given query requirement. As shown below, the entity-relationship diagrams can greatly help the design of NoSQL column store databases. Consider the example in Fig. 8. The question is "What NoSQL Schema options do we have in this entity-relationship diagram?".

For the1:N relationship between *Category* and *Posting*, two reasonable query requirements could be:

Query 1: For each category, show all of the postings.

Query 2: Display the postings with their categories.

Fig. 9 shows the relationship between *Category* and *Posting*. The query "for each category, show all the postings" is handled by embedding the posting data in Category, using the column family and nesting.

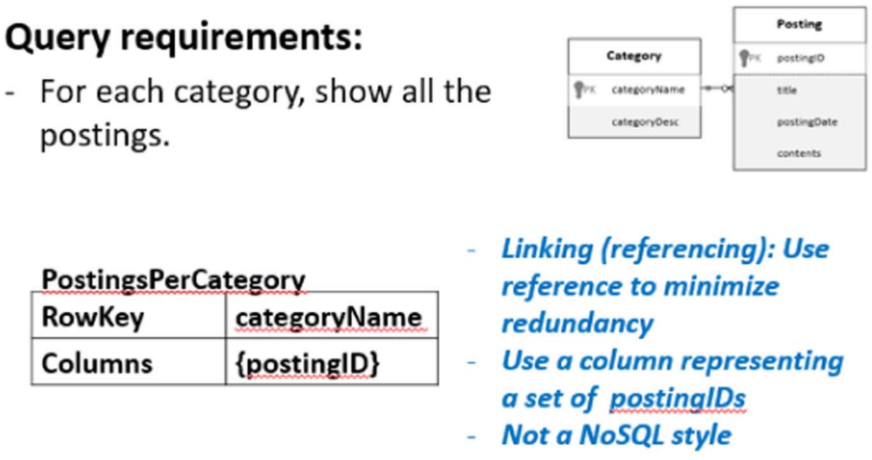
A linking solution is shown in Fig. 10.

The second query, "display posting with its category" is shown in Fig. 11. The category data is embedded within the posting table with denormalization.

If we keep both tables, maintaining consistency between two tables is critical. NoSQL could end up having many more denormalized tables when trying to satisfy all of the query requirements as shown in Fig. 12.

For the M:N relationship between Posting and Tag, there are two query requirements: for each tag, show all the related postings; and show the tags of each posting. There are several different ways to model this to facilitate the query requirements. If there is an attribute of the M:N relationship, the NoSQL model could also be further changed.

Several important implications arise. For each access path, one needs to create an aggregate table whose row key begins with the attribute used in searching. For 1:N relationships, a denormalized structure is commonly used. In M:N relationships, there are multiple ways of creating aggregates: entity-focused, relationship-focused, one-way access, two-way accesses, reverse-timestamp, etc. If an aggregate has a transitive dependency within the aggregate, then a Column family can be used inside the aggregate. If a subaggregate is a repeating group, a nested structure can be used. The most important fact is that entity-relationship diagrams are still useful for designing a NoSQL schema.


Query requirements:

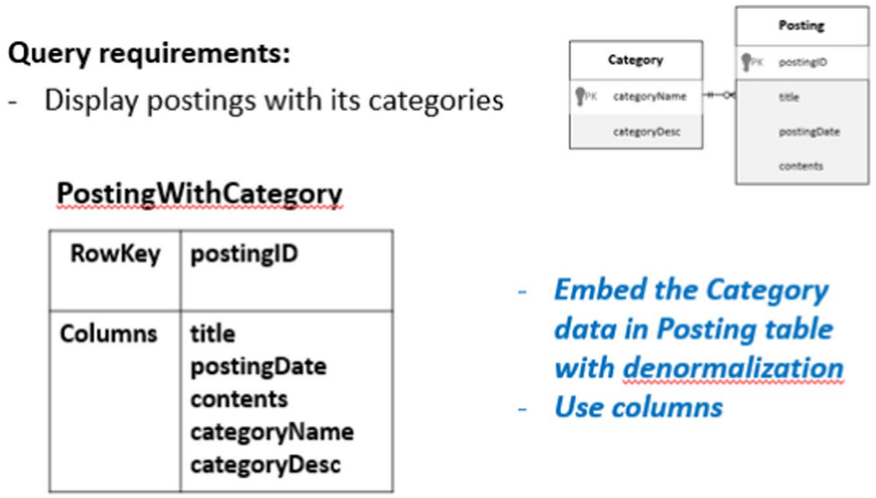
- For each category, show all the postings.

PostingsPerCategory

RowKey	categoryName
Columns	{postingID}

- *Linking (referencing): Use reference to minimize redundancy*
- *Use a column representing a set of postingIDs*
- *Not a NoSQL style*

Fig. 10. 1:N Relationship with Linking for Query 1.



Query requirements:

- Display postings with its categories

PostingWithCategory

RowKey	postingID
Columns	title postingDate contents categoryName categoryDesc

- *Embed the Category data in Posting table with denormalization*
- *Use columns*

Fig. 11. 1:N Relationship with Embedding for Query 2.

Column stores supports diverse modeling structures and have schema flexibility (dynamic schema). However, the design is dependent on access paths. There is data redundancy in multiple tables, which requires careful integrity management at the coding level. The number of tables in a Column Store could be much larger than that in a relational database, when trying to satisfy all of the query requirements. Data integrity maintenance is at the coding level. No complex query processing is available in most column stores.

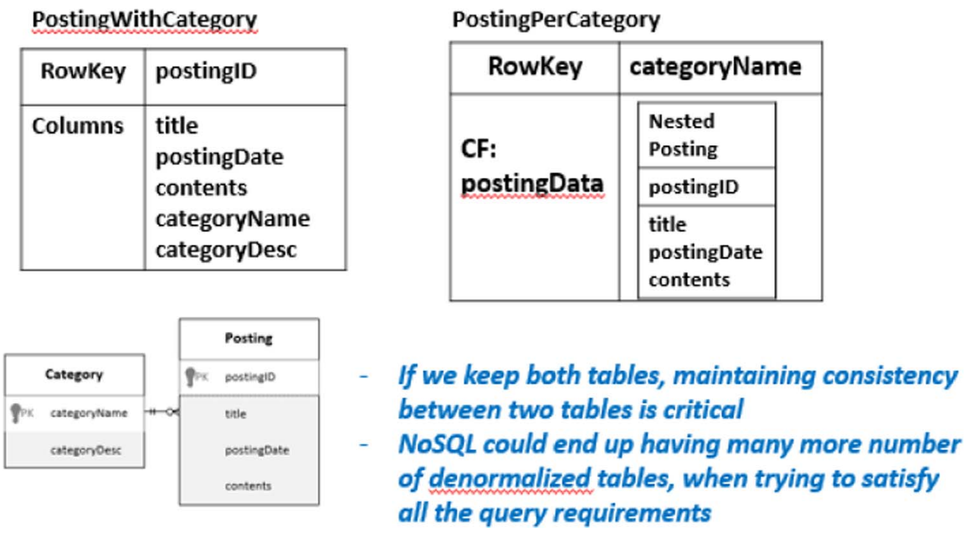
4.2.3. Document store

A document store is a collection of key-value stores where the value is a document, such as JSON, BSON, etc. Each document has a unique key which is assigned to retrieve the document. Any collection of data can be stored such as nested structures, maps, collections, and scalar values, There can be secondary indexes to access a component of a document. Document stores are found in MongoDB, CouchDB, and others.

From a modeling perspective, document stores are good for semi-structured data. A document store supports aggregates and denormalized structures. All 1:1 and 1:N relationships can be embedded in a single document. Referencing static relationships, such as that found between a book and a publisher, is also possible. The rich modeling structures (nested structure, variable array) of document stores are more powerful than column stores, and good for sparsely populated data sets. However, there is a lack of standardization for modeling structures and a query language.

4.2.4. Graph database

Graph databases are based upon graph theory (set of nodes, edges, and properties) and useful for *inter-connected relationship*



PostingWithCategory

RowKey	postingID
Columns	title postingDate contents categoryName categoryDesc

PostingPerCategory

RowKey	categoryName	
CF: postingData	Nested Posting	
	postingID	
	title postingDate contents	

- *If we keep both tables, maintaining consistency between two tables is critical*
- *NoSQL could end up having many more number of denormalized tables, when trying to satisfy all the query requirements*

Fig. 12. Redundant representations for a 1:N Relationship for Query 1 and Query 2.

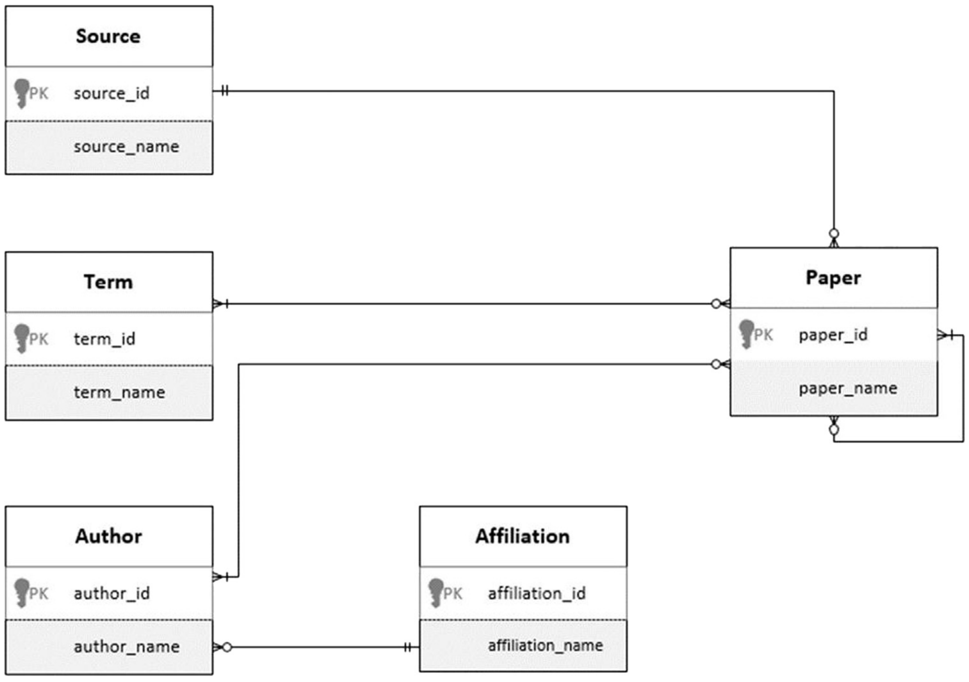


Fig. 13. Entity-relationship model for GIBIR.

data such as communication patterns, social networks, and biographical interactions. They allow one to ask deeper, more complex, questions and express queries as traversals. It is difficult, however, to distribute components of a graph among a network of servers as graphs become larger. It may more accurately be described as non-relational, not really NoSQL. Examples are Neo4j, Titan, and OrientDB.

As an example, the *Graph-based Interactive Bibliographic Information Retrieval System* (GIBIR) was developed to deal effectively with papers, authors, citations, affiliation, sources, terms, etc. by Zhu et al. [46]. GIBIR is implemented in Neo4j, a highly scalable, open source graph database that supports ACID. Graphs are stored in disks in a custom binary format. The entity-relationship model [4] for GIBIR is shown in Fig. 13, illustrating the usefulness of a conceptual model representation for a graph database.

The corresponding graph schema of Fig. 13 is shown in Fig. 14, representing typical bibliographic entities and relationships.

*Performance***:** With respect to performance, the graph model-based system outperforms the relational model-based system in retrieving bibliographic information because the graph system traverses instead of making joins. The relational database performed better in executing queries with only two nodes. However, as queries became complex, the graph database outperformed the relational one.

Conceptual Modeling: An entity-relationship diagram is readily translated into a Property Graph Model, making a conceptual model for graph databases necessary. It helps to understand which entities can be logically connected to which other entities. Graph databases support only binary relationships. Graph modeling is much easier than for a relational data model because real world objects are explicit in terms of connections.

Assessment: The graph database is good for applications with complex relationships. It is faster than relational models when multi-nodes are involved. However, the Graph databases are *not* cluster-friendly. Big graphs may not fit into memory on a single node. Graph partitioning causes many problems. Inter-node communications / messaging are critical. Currently, there are no standard APIs or query languages for graph databases.

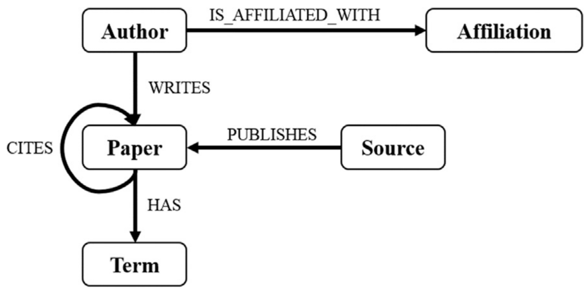


Fig. 14. The property graph of GIBIR.

4.3. Problems and use cases of NoSQL databases

NoSQL is not good for OLTP (online transaction processing) that requires ACID properties and frequent updates. Eventual consistency is acceptable in some applications where performance is more important than consistency. Examples include search engines that need to quickly display different search results or recommender systems that could display different review results. NoSQL products, however, are rapidly evolving. Some systems, such as FoundationDB and OrientDB, claim they support ACID.

The impact of NoSQL databases can be appreciated by their many use cases. Mobile applications, Internet of Things, and real-time big data analytics use Key-Value, column store and document store databases. Multi-sourced structured and unstructured document management systems use document stores. Personalization, content management systems, eLearning, and catalog systems use column store and document store databases. Patient health and prescription management systems use Key-Value databases, as do real-time gaming. Massive marketing services which strive to capture a holistic view of customers use column stores. Data management from social networks uses graph databases. Graph databases are also employed for master data management and version management systems, and complex relationship management in drugs.

5. Conceptual modeling and big data management

Conceptual modeling has, since its beginning, focused on the organization of data [4,8,24]. It requires the construction of a conceptual representation of the application domain of an information system [41,43]. For big data, the importance of conceptual modeling can be considered from both technical and management perspectives.

Conceptual modeling is intended to describe the semantics of software applications. Conceptual modelers describe structural models, behavior or functional models, as well as interactions and user interfaces. Such models enable clients and analysts to understand one another, thereby facilitating useful communication among them (www. conceptualmodeling.org). Conceptual modeling is useful in abstracting and decomposing complex concepts and can assist in understanding, managing, architecting, and teaching database technologies. Techniques for conceptual modeling must address issues related to handling data integration, big data warehousing, cloud, metadata, and curation. Embley and Liddle [8] argue that big data, as well as regular data, requires careful modeling to ensure its appropriate use as representative of the real world. Scientists and designers trained in conceptual modeling must be "talented thinkers" in the sense that they are capable of abstracting, representing, extracting, managing, analyzing, and visualizing results.

Although conceptual modeling is useful for modeling big data, conceptual modelers, still, must overcome a number of challenges to ensure successful big data projects:

- *Understand/utilize/apply life cycles* – understand traditional life cycles for regular databases, as well as big data life cycles that address the big data challenges.
- *Identify big data requirements and connect business goals and technologies* – the volume of big data is extremely large, so conceptual modelers must be able to identify the requirements that address the business goals and to select the right big data technologies for the problem and requirements at hand.
- *Identify use cases, build architect, and select platforms* – these can be helpful in the creation of appropriate platforms.
- *Lead big data projects* – conceptual modelers could play the roles of the Chief Data Office (CDO) who is responsible for the management of big data projects.
- *Perform high-level analysis* — before embarking on a detailed big data analytics project, conceptual modelers should be able to perform a high level analysis to understand the return-on-investment of the analytics project.
- *Evaluate big data projects* – as any data management project, big data projects must be evaluated for their feasibility, usefulness, and quality.
- *Help manage the veracity of big data* – continued recognition is needed of the usefulness of the veracity characteristic of big data.
- *Domain analysis, governance and Master Data Management* – generic activities associated with traditional, as well as big data, projects, must be managed.

These activities collectively help derive the success of big data projects. Conceptual modeling, in general, contributes to the 5Vs of big data as follows.

- **Volume**: a physical characteristic, but conceptual modeling can organize, identify and describe important data and metadata.
- **Velocity**: data must be filtered, but conceptual modeling can help to extract important data.
- **Variety**: conceptual modeling can model the variety, hierarchies, and networks of data, integrate the data, and address the resulting big data warehousing issues.
- **Veracity**: conceptual modeling can check for quality, completeness, and consistency.
- **Value**: conceptual models can manage big data projects, including analyzing them to extract value and evaluate the results.

Thus, conceptual modeling can play an important role in big data initiatives.

5.1. Challenges

There are many challenges and problems associated with big data project management, the most notable of which are described below.

Data-driven culture: The data must be considered objectively, without relying on intuition. This is evident from the successful supply-chain management of companies such as Apple, Google, and Wal-mart.

Business goals: Tie the project to the goals of the business. Ensure that the big data project matches well with the needs of the business application.

Recruit proven analytical talent: Acquire the needed analytical talent. Innovations, as identified by such talent, can come from performing diagnostics, predictive, and prescriptive analytics.

Understand big data solutions: Solutions can involve Hadoop ecosystems, Spark, NoSQL, cloud, in-memory computing, and data virtualization.

Security: As with regular data management projects, data security is always a major challenge.

Leadership: Leadership is required throughout an entire data analytic life cycle.

5.2. Data analytics lifecycle

The cross-industry standard process for data mining, CRISP-DM [3] consists of several components. *Business understanding* includes comprehending what questions need to be solved and what metrics are appropriate for the evaluation. *Data understanding* requires the identification of the appropriate data resources and tools needed. *Data preparation* involves acquiring the data, cleaning and transforming the data, and verifying the data quality. *Model planning* identifies the methods, techniques, and workflows. Key variables are selected and the correlations between them identified. Evaluation consists of evaluating the results against metrics and communicating the results. Finally, *deployment* is the integration of analytics procedures into management dashboards and operating systems.

A data analytics life cycle is proposed in Fig. 15 [33]. It proceeds from business strategy to questions, data acquisition, solution generation, evaluation, and monitoring.

The Chief Data Officer (CDO) has emerged as someone responsible for managing the data assert for purposes of governance and innovation. The Chief Data Officer must be able to provide vision and strategy for data management initiatives, and carry out other duties as shown in Fig. 16 [33]. These duties cover activities from conceptual analysis to deployment, which highlights the importance of acquiring conceptual modeling knowledge.

6. Discussion

Big data problems are also problems in computer science. Addressing them requires, first, the creation of a business case for the need for big data analytics. The data must be examined to identify which data must be merged and combined from multiple data

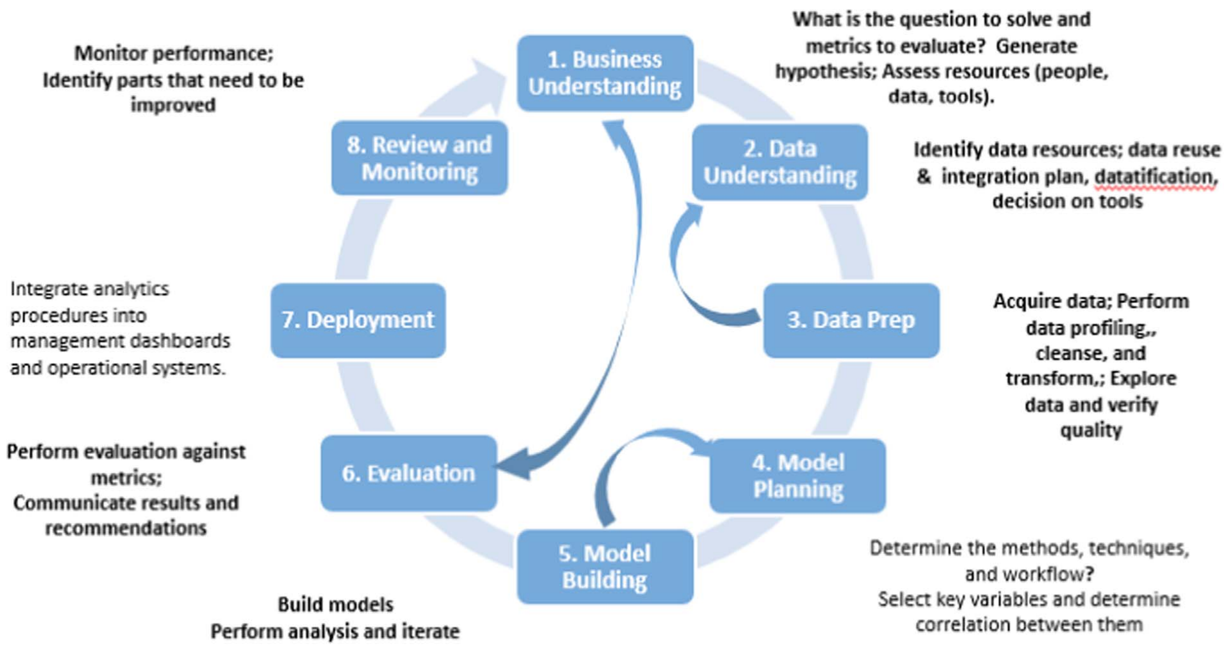

1. Business Understanding
What is the question to solve and metrics to evaluate? Generate hypothesis; Assess resources (people, data, tools).
2. Data Understanding
Identify data resources; data reuse & integration plan, datatification, decision on tools
3. Data Prep
Acquire data; Perform data profiling,, cleanse, and transform,; Explore data and verify quality
4. Model Planning
Determine the methods, techniques, and workflow? Select key variables and determine correlation between them
5. Model Building
Build models Perform analysis and iterate
6. Evaluation
Perform evaluation against metrics; Communicate results and recommendations
7. Deployment
Integrate analytics procedures into management dashboards and operational systems.
8. Review and Monitoring
Monitor performance; Identify parts that need to be improved

Fig. 15. Data Analytics Life Cycle.

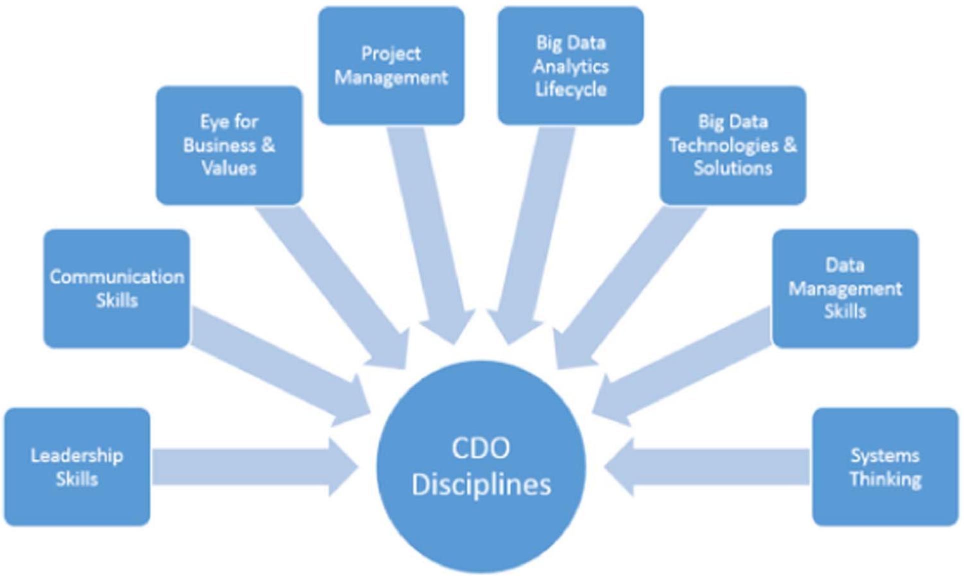


Fig. 16. Chief Data Officer.

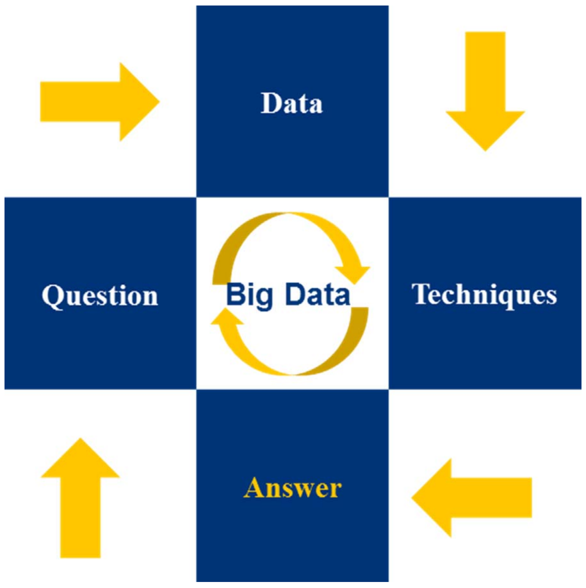


Fig. 17. Big Data Management Initiatives.

sources. The desired output data must be assessed to identify how to compute the desired results. The results obtained must then be interpreted and evaluated in an attempt to find new aspects in knowledge discovery. The abstraction of these big data management issues is illustrated in Fig. 17.

Big data technologies are complex and evolving. They are needed to address many issues related to web, mobile, social, cloud, and big data analytics, as these continue to mature. Hadoop ecosystems are also evolving. However, despite the hype, Hadoop has had a rather slow acceptance rate. Spark, however, is rapidly dominating real-time parallel processing. NoSQL systems require standard query languages and discipline for modeling. In-memory computing is widely used in NoSQL, with NewSQL useful for analytics. Although Hadoop and enterprise data warehouses (EDW) currently co-exist, enterprise data warehouses are evolving into an integrated data management platform. Emerging social and technology developments continue to drive big data such as the Internet of things (IoT) [45], and wide-scale adoption of personal and professional user-generated content on social media, and other outlets, that can be used for sentiment and additional forms of analysis. For NoSQL databases, specific technical research issues include: design patterns, two phase commit, visualization, partitioning of graphs using data mining and conceptual modeling in graph databases, and standard query languages.

From a management perspective, the big data life cycle management must concentrate on the 5Vs. Research questions include:

How can one (semi)-automatically capture meta-data within the context of big data life cycles (acquisition, transfer, collection, distribution, integration, preparation, analysis, visualization, governance, migration, maintenance, curation)? What are the essential minimum metadata sets needed to support big data life cycle management? How should one label, manage, track, and apply metadata to real-world projects? What are the best practices for doing so?

There are many ongoing, and interesting real-world research projects that are based on effective use of big data. Among them are topics related to smart cities, smart health, smart aging, and cognitive computing. Some of the systems being built to support these are difficult to program, making adoption slow. Others have more quickly become adopted (e.g. using mobile apps).

Leading advances in healthcare include, for example, using Watson (https://www.ibm.com/watson/health/) for patient care. Smart aging mitigates the effect of aging for the elderly by using medical, computer, and communication technologies. The results are opportunities for healthy aging. Using diverse wearable technologies and various sensors for monitoring patient activities and elderly wellness could generate large amounts of diverse lifelog data such as video streams, images, audio, and text data. They include notions from the data-driven paradigm which “nudges” people towards healthy choices such as: early detection of declining good habits (e.g., nutrition, sleeping, keeping active); early intervention to suggest healthier alternatives; and incorporation of preferences in insurances, treatments, social services, and activities. Conceptual modeling could play significant roles in modeling, annotating, integrating, and extracting relevant data and metadata for analytics and curation.

Projects involving big data can have a large impact on project management requirements, especially given their potential to influence operations in management and society by, for example, dealing with larger data sets than in the past, and acting upon novel data analysis results or pattern extraction results. In the world of big data, conceptual modeling activities are useful and valid. Big data requirements need to be identified, understood, and represented. NoSQL database modeling differs from traditional relational database modeling, requiring modelers to understand both big data concepts and design architectures.

Conceptual modeling efforts are needed to understand concepts related to big data, as well as to create a design architect. The Vs of big data provide a useful way to consider specific implications for conceptual modeling. With respect to work on variety, veracity, and value, conceptual modeling has distinctive roles to play. To deal with the variety of data, NoSQL, design patterns, methodologies, and tools are needed. The veracity of data results in the need to solve data and system quality issues. Quality management is challenging, but crucial. For the value to be realized, the correct architectures and solutions must be modeled that can generate actionable knowledge to improve business processes. There are also long-standing problems related to data quality [42] and semantics [32,39].

As the focus on both technical tools and skills are developed for handling big data, equally important are the modeling techniques to represent how big data might be used and the management capabilities required to extract value from big data. Many opportunities are emerging, so conceptual modelers can expand their scope. To empower conceptual modelers for data analytics, they might become citizen data scientist (e.g., [20]), engage in a management role as a Chief Data Officer (to identify business opportunities and datafication) or as a Big Data Architect (to select platforms, design architectures, and technologies).

NoSQL DBs are still in an early phase of maturation. For them to become accepted and popular, good methodologies and tools are needed for design, integrity, and consistency. Standard query languages are also required. Relational databases will, no doubt, continue to be used for transaction-based systems. NoSQL and relational systems will co-exist. NoSQL is finding wide-spread use in conjunction with search engines; web-based systems; real-time, cloud, and mobile applications; low-cost initial engagement; the Internet of Things; and others.

7. Conclusion

Big data has the potential to make large impacts on business and society, some of which are already being realized. This paper has reviewed recent developments in big data technologies and discussed challenges of big data, presented in terms of five “Vs” along with potential approaches to addressing them, recognizing that the value of big data is the most difficult to attain. From reviewing big data technologies, potential conceptual modeling contributions were identified for how they can support efforts and opportunities for innovative solutions to complex problems.

Finally, it is important to teach big data technologies to the next generation of computer scientists, conceptual modelers, and management information systems professionals (database administrators, systems analysts, etc.) [33]. Big data technologies need to be understood, including software and hardware approaches to dealing with big data, as well as approaches to big data analytics using sophisticated and automated tools. Collaboration of the efforts of researchers in the areas of conceptual modeling and database management and development are required to address the magnitude of issues that arise as big data, the Internet of Things, and many other interesting, and timely, applications evolve.

Acknowledgement

Thanks to Youngjun Zhu, PhD Student, Drexel University, Jane Greenberg, Director of MRC, Drexel University, Thomas Baley, and Wolfgang Maass. This research was supported in part by Drexel University and by the J. Mack Robinson College of Business, Georgia State University.

References

[1] A. Abbassi, A. Sarker, R.H.L. Chiang, Big data research in information systems: toward an inclusive research agenda, J. Assoc. Inf. Syst. 17 (2) (2016) (i-xxxii).
[2] N. Abdullah, S.A. Ismail, S. Sophiayati, S.M. Sam, Data quality in big data: a review, Int. J. Adv. Softw. Comput. Appl. 7 (3) (2015).
[3] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, & R. Wirth, CRISP-DM 1.0 Step-by-step data mining guide, 2000.
[4] P.P.S. Chen, The entity-relationship model—toward a unified view of data, ACM Trans. Database Syst. ((TODS)) 1 (1) (1976) 9–36.
[5] H. Chen, R.H. Chiang, V.C. Storey, Business intelligence and analytics: from big data to big impact, MIS Q. 36 (4) (2012) 1165–1188.
[6] E.F. Codd, A relational model of data for large shared data banks, Commun. ACM 13 (6) (1970) 377–387. http://dx.doi.org/10.1145/362384.362685.
[7] V. Dhar, Data science and prediction, Commun. ACM 56 (12) (2013) 64–73.
[8] D.W. Embley, S.W. Liddle, Big data—conceptual modeling to the rescueConceptual Modeling, Springer Berlin Heidelberg, 2013, pp. 1–8.
[9] Gartner, ⟨https://www.gartner.com/doc/2191415/big-data-strategy-components-business⟩.
[10] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions, Future Gener. Comput. Syst. *29* (7) (2013) 1645–1660.
[11] F.H. Hsu, Behind Deep Blue: Building the Computer that Defeated the World Chess Champion, Princeton University Press, 2002.
[12] Kalakota, R. (2012). Big Data Infographic and Gartner 2012 Top 10 Strategic Tech Trends.
[13] M. Khan, M. Uddin, and N. Gupta, Seven V's of Big Data: Understanding Big Data to extract Value, in: Proceedings of 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1), 2014.
[14] G. King, Ensuring the data-rich future of the social sciences, Science (2011) 719–721.
[15] R. Kitchin, T.P. Lauriault, Small data in the era of big data, Geo. J. (2014) 1–13.
[16] P. Hitzler, K. Janowicz, Linked data, big data, and the 4th paradigm, Semant. Web. 4 (3) (2013) 233–235.
[17] A. Lakshman, P. Malik, Cassandra - a decentralized structured storage system, ACM SIGOPS Oper. Syst. Rev. 44 (2) (2010) 35. http://dx.doi.org/10.1145/1773912.1773922.
[18] Laney, D., "3D Data Management: Controlling Data Volume, Velocity and Variety" (PDF). Gartner, 2001.
[19] M. Lin, H.C. Lucas Jr., G. Shmueli, Research Commentary – Too Big to Fail: large Samples and the p-Value Problem,, Inf. Syst. Res. 24 (4) (2013) 906–917.
[20] R. Lukyanenko, J. Parsons, Y. Wiersma, Citizen science 2.0: data management principles to harness the power of the crowdService-Oriented Perspectives in Design Science Research, Springer Berlin Heidelberg, 2011, pp. 465–473.
[21] V. Mayer-Schönberger, K. Cukier, Big data: A Revolution that Will Transform how We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.
[22] A. McAfee, E. Brynjolfsson, Big Data: the management revolution, Harv. Bus. Rev. 90 (10) (2012) 60–68.
[23] P. Mell, T. Grance, The NIST definition of cloud computing, Natl. Inst. Stand. Technol. 53 (6) (2009) 50.
[24] C.W. Mineau, R. Missaoui, R. Godin, Conceptual modeling for data and knowledge management, Data Knowl. Eng. 33 (2) (2000) 137–168.
[25] J. Pokorny, NoSQL databases: a step to database scalability in web environment, Int.J. Web Inf. Syst. 9 (1) (2013) 69–82.
[26] A. Ribeiro, A. Silva, A.R. da Silva, Data modeling and data analytics: a survey from a big data perspective, J. Software Eng. Appl. 8 (12) (2015) 617.
[27] P.A. Rubin, Big data in small bites, Decis. Line (2015) 10–12.
[28] M. Russo, Redis, from the Ground Up. Retrieved November 26, 2015, from ⟨http://blog.mjrusso.com/2010/10/17/redis-from-the-ground-up.html⟩, (2010, October 17).
[29] P.J. Sadalage, M. Fowler, NoSQL distilled: a brief guide to the emerging world of polyglot persistence, Pearson Education, 2012.
[30] SAP - In-Memory Computing, New Reality of Real Time With Launch of SAP® High-Performance Analytic Appliance. Retrieved November 26, 2015, from ⟨http://global.sap.com/corporate-en/news.epx?Pressid=14457⟩, (2010, December 1).
[31] K.D. Schewe, B. Thalheim, Conceptual modelling of web information systems, Data Knowl. Eng. 54 (2) (2005) 147–188.
[32] K.D. Schewe, B. Thalheim, Semantics in Data and Knowledge Bases, Springer Berlin Heidelberg, 2008, pp. 1–25.
[33] I.-Y. Song, Y. Zhu, Big data and data science: what should we teach?, Expert Syst. 33 (4) (2016) 364–373.
[34] M. Stonebraker, New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps | blog@CACM | Communications of the ACM. Retrieved November 26, 2015, from ⟨http://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/fulltext⟩, June 16 2011.
[35] V.C. Storey, Relational database design based on the entity-relationship model, Data Knowl. Eng. 7 (1991) 47–83.
[36] V.C. Storey, The Real Challenges of Big Data Decision Line, January 2016.
[37] V.C. Storey, J.C. Trujillo, S.W. Liddle, Conceptual modeling: themes, topics, and introduction to the special issue, Data Knowle. Eng. (2015).
[38] T.J. Teorey, D. Yang, J.P. Fry, A logical design methodology for relational databases using the extended entity-relationship model, ACM Comput. Surv. ((CSUR)) 18 (2) (1986) 197–222.
[39] B. Thalheim, Syntax, semantics and pragmatics of conceptual modelling.Natural Language Processing and Information Systems, Springer Berlin Heidelberg, 2012, pp. 1–10.
[40] A. Vasiliev,. World of the NoSQL Databases. Retrieved November 26, 2015, from ⟨http://leopard.in.ua/2013/11/08/nosql-world/⟩, (November 8) 2013.
[41] Y. Wand, R. Weber, Research commentary: information systems and conceptual modeling—a research agenda, Inf. Syst. Res. 13 (4) (2002) 363–376.
[42] R.Y. Wang, V.C. Storey, C.P. Firth, A framework for analysis of data quality research, IEEE Trans. knowl. data eng. 7 (4) (1995) 623–640.
[43] R. Weber, Research review paper conceptual modelling and ontology: possibilities and pitfalls, J Database Manag 2003a, 14(3) (1-20) (2003).
[44] Wikibon (2015), ⟨http://premium.wikibon.com/executive-summary-big-data-vendor-revenue-and-market-forecast-2011-2020/⟩, 2013.
[45] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, Internet Things J., IEEE 1 (1) (2014) 22–32.
[46] Y. Zhu, E. Yan, I.-Y. Song, The use of a graph-based system to improve bibliographic information retrieval: system design, implementation, and evaluation, J. Assoc. Inf. Sci. Technol. 68 (2) (2016) 480–490.

Source URLS

Source:
⟨http://practicalanalytics.wordpress.com/2012/08/20/innovation-and-big-data-in-corporations-a-roadmap/⟩.
Source:
⟨http://www.datanami.com/2015/08/26/why-gartner-dropped-big-data-off-the-hype-curve⟩.
Source:
⟨http://radar.oreilly.com/2011/08/building-data-startups.html⟩.
Source:
⟨http://futureanalytics.blogspot.com/⟩.
Source:
⟨http://radar.oreilly.com/2011/08/building-data-startups.html⟩.
,
⟨http://cumuluspartners.com/cumulus/file.axd?file=images%2FReal+Time+Big+Data+Analytics.pdf⟩.
Source:
Real-Time Big Data Analytics: Emerging Architecture by Mike Barlow, O'Reilly, 2013.
Tailor et al.,
H. Tailor, S. Choudhary, V. Jain, Rise of NewSQL.

Aslett, 2011
M. Aslett, "How Will The Database Incumbents Respond To NoSQL And NewSQL?". 451 Group (published 2011-04-04), 2011.
Gartner,
Gartner, Retrieved from ⟨http://timoelliott.com/blog/2013/04/why-in-memory-computing-is-cheaper-and-changes-everything.html⟩.
Gartner,
Gartner, Taxonomy, Definitions and Vendor Landscape for In-Memory Technologies.
Greeley and Ebbels,
B. Greeley and D. Ebbels, Enabling Mobile App Developers with NoSQL in the Cloud.
Source:
⟨http://searchcloudcomputing.techtarget.com/definition/hybrid-cloud⟩.
Davenport,
R. Davenport, ETL vs ELT A Subjective View.
Source:
⟨http://www.dataversity.net/data-ed-slides-trends-in-data-modeling/⟩.
Source:
⟨http://www.forbes.com/sites/teradata/2015/09/15/is-data-modeling-dead/⟩.
Gartner,
Gartner, A Tour of NoSQL in Eight use Cases.
Source:
⟨https://developer.ibm.com/bluemix/2014/10/17/building-java-ee-app-ibm-bluemix-using-watson-cloudant/⟩.
Source:
⟨http://neo4j.com/blog/why-the-most-important-part-of-facebook-graph-search-is-graph/⟩.
Tang et al., 2008
J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, & Z. Su, . Arnetminer: extraction and mining of academic social networks. in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining pp. 990–998. ACM, August 2008.
Source:
⟨http://bigdatawg.nist.gov/_uploadfiles/M0397_v1_2395481670.pdf⟩.
Source:
DB-Engines, ⟨http://db-engines.com/en/ranking⟩.
Brewer,
E. Brewer, InfoQ, ⟨http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed⟩.
Brewer,
E. Brewer, InfoQ, ⟨http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed⟩.
Source:
⟨http://stackoverflow.com/questions/11292215/where-does-mongodb-stand-in-the-cap-theorem⟩.
Source:
⟨http://blog.prabasiva.com/2012/04/09/traditional-edw-vs-big-data/⟩.
Chapman et al., 2000
P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, & R. Wirth, CRISP-DM 1.0 Step-by-step data mining guide, 2000.
Source:
⟨https://en.wikipedia.org/wiki/FLOPS⟩.
Source:
⟨http://www.dataversity.net/data-ed-slides-trends-in-data-modeling/⟩).
Source:
⟨http://techtech.chayan.dyndns-home.com/wordpress/?M=201402⟩.

Veda C. Storey is the Tull Professor of Computer Information Systems and Professor of Computer Science at the J. Mack Robinson College of Business, Georgia State University. Her research interests are in conceptual modeling, intelligent information systems, big data, and design science research. She has published in MIS Quarterly, Information Systems Research, ACM Transactions on Data Bases, IEEE Knowledge and Data Engineering, and Data and Knowledge Engineering. She has served as a guest senior editor for MIS Quarterly for special issues in design science research and business intelligence and data analytics. She currently serves on the Steering Committee of the International Conference on Conceptual Modeling where she is an ER Fellow.

Dr. Il-Yeol Song is professor in the College of Computing and Informatics of Drexel University. He served as Deputy Director of NSF-sponsored research center on Visual & Decision Informatics (CVDI) between 2012-2014. He is an ACM Distinguished Scientist and an ER Fellow. He is the recipient of 2015 Peter P. Chen Award in Conceptual Modeling. His research interests include conceptual modeling, data warehousing, big data management, data analytics, CRM, and smart aging. Dr. Song published over 210 peer-reviewed papers in these areas. He is a co-Editor-in-Chief of Journal of Computing Science and Engineering (JCSE) as well as an Area Editor of Data & Knowledge Engineering (DKE). He won the Best Paper Award in the IEEE CIBCB 2004. He won four teaching awards from Drexel, including the most prestigious Lindback Distinguished Teaching Award. Dr. Song served as the Steering Committee chair of the ER conference between 2010-2012. He served as a chair/co-chair of over 20 international conferences and workshops, including DOLAP 1998-2015, CIKM 1999, ER 2003, FP-UML2006, DaWaK2007-08, DESRIST2009, CIKM 2009, MoBiD 2013-2015, and ER2016. He delivered a keynote speech on big data at the First Asia-Pacific iSchool conference in 2014, ACM SAC 2015 conference, ER2015 Conference, EDB 2016 conference, and A-LIEP 2016 conference.