History of Database Applications

Hehua Chi
University of Rochester
Wegmans Hall 1210, Rochester, NY 14620
hchi3@ur.rochester.edu

Yihe Yang
University of Rochester
Wegmans Hall 1210, Rochester, NY 14620
yyang107@ur.rochester.edu

ABSTRACT
In this paper, we explored the history of database applications which are all about three revolutions in database technologies. The first revolution was driven by the emergence of the electronic computer. In the 20 years following the widespread adoption of electronic computers, a range of increasingly sophisticated database systems emerged. The second revolution was the emergence of the relational database. Shortly after the definition of the relational model in 1970, almost every significant database system shared a common architecture. The three pillars of this architecture were the relational model, ACID transactions, and the SQL language. However, starting around 2008, the third revolution has resulted in an explosion of non-relational database alternatives driven by the demands of modern applications that require global scope and continuous availability. An explosion of new database systems occurred: key-value database, document database, graph database, column database or even SSD and In-memory database. The next generation databases will be NoSQL, NewSQL and big data platforms [1].

1. INTRODUCTION
Wikipedia defines a database as an "organized collection of data." Although the term database entered our vocabulary only in the late 1960s, collecting and organizing data has been an integral factor in the development of human civilization and technology. Books, libraries and other indexed archives of information represent preindustrial equivalents of modern database systems [1].

The emergence of electronic computers following the Second World War represented the first revolution in databases. The development of indexing methods such as ISAM (Index Sequential Access Method) and similar indexing structures powered the first electronic databases. However, there was no Database Management Systems (DBMS) which can minimize programmer overhead and ensure the performance and integrity of data access routines [1].

By the early 1970s, two major models of DBMS were competing for dominance. The network model was formalized by the CODASYL standard and implemented in databases such as IDMS, while the hierarchical model provided a somewhat simpler approach as was most notably found in IBM's IMS (Information Management System) [1].

However, these systems had several notable drawbacks. First, the navigational databases were extremely inflexible in terms of data structure and query capabilities. And it was extremely difficult to add new data elements to an existing system. Second, the database systems were centered on record at a time transaction processing. Query operations, especially the sort of complex analytic queries that we today associate with business intelligence, required complex coding [1].

Arguably, no single person has had more influence over database technology than Edgar Codd. He harbored significant reservations about their design. In particular, he considered the following restrictions [1, 2]:

1. Existing databases were too hard to use. Databases of the day could only be accessed by people with specialized programming skills.
2. Existing databases lacked a theoretical foundation. Codd's mathematical background encouraged him to think about data in terms of formal structures and logical operations.
3. Existing databases mixed logical and physical implementations.

Codd published an internal IBM paper outlining his ideas for a more formalized model for database systems, which then led to his 1970 paper "A Relational Model of Data for Large Shared Data Banks." This classic paper contained the core ideas that defined the relational database model that became the most significant—almost universal—model for database systems for a generation [1, 2].

The relational model does not itself define the way in which the database handles concurrent data change requests. These changes—generally referred to as database transactions. Jim Gray defined the most widely accepted transaction model in the late 1970s. This soon became popularized as ACID transactions: Atomic, Consistent, Independent, and Durable [2].

However, the restriction on scalability beyond a single data center implied by the ACID transaction model has been a key motivator for the development of new database architectures. The difference in application architectures between the client-server era and the era of massive web-scale applications created pressures on the relational database that could not be relieved through incremental innovation [1].

A sort of database explosions occurred in the years 2008 - 2009: literally dozens of new database systems emerged in this short period. Especially in late 2009, the term NoSQL quickly caught on as shorthand for any database system that broke with the traditional SQL database [3]. By 2011, the term NewSQL became popularized as a means of describing this new breed of databases that, while not representing a complete break with the relational model, enhanced or significantly modified the fundamental principles. Finally, the term Big Data burst onto mainstream consciousness in early 2012. Although the term refers mostly to the new ways in which data is being leveraged to create value, we generally understand "Big Data solutions" as convenient shorthand for technologies that support

large and unstructured datasets such as Hadoop. NoSQL, NewSQL, and Big Data are in many respects vaguely defined, overhyped, and overloaded terms. However, they represent the most widely understood phrases for referring to next-generation database technologies [1].

The remaining of this paper is organized as follows: the history of database applications are reviewed in Section 2. Two promising sub-areas of future database systems are introduced in Section 3. The considerations and requirements for choosing appropriate database systems in different applications are summarized in Section 4. Finally, the conclusion is in Section 5.

2. HISTORY OF DATABASE APPLICATIONS

2.1. Timeline of database development

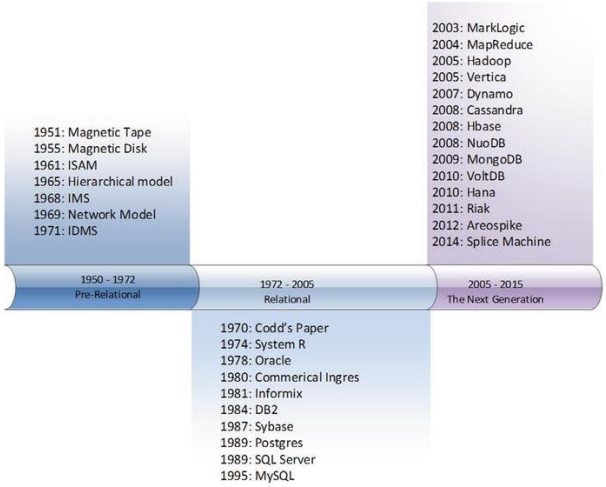

Figure 1. Illustrating three major eras in database technology [1].

The first revolution was driven by the emergence of the electronic computer and then many pre-relational databases have sprung up like mushrooms. The second revolution was driven by a classic paper contained the core ideas that defined the relational database model that became the most significant - almost universal - model for database systems for a generation [5]. The third revolution for next generation databases has resulted in an explosion of non-relational database alternatives to meet the era of massive web-scale and big data applications [4]. Figure 1 illustrates three major eras in database technology. In this section, we'll provide an overview of these three waves of database technologies and discuss the tendency forces leading to today and future's next generation databases.

2.1.1 The pre-relational database system

Early database systems enforced both a schema (a definition of the structure of the data within the database) and an access path (a fixed means of navigating from one record to another). Table 1 illustrates the pre-relational database system development. By the early 1970s, two major models of DBMS were competing for dominance: the network model and the hierarchical model [1].

Table 1. Pre-relational database system development

Year	Pre-Relational Database system
1951	Magnetic tape
1955	Magnetic Disk
1961	ISAM
1965	Hierarchical model
1968	IMS
1969	Network Model
1971	IDMS

2.1.2 Relational database system

The intricacies of relational database theory, at its essence, describes how a given set of data should be presented to the user, rather than how it should be stored on disk or in memory. A row in a table should be identifiable and efficiently accessed by a unique key value, and every column in that row must be dependent on that key value and no other identifier. Arrays and other structures that contain nested information are, therefore, not directly supported [1]. Table 2 illustrates the development of relational database systems. While each of these systems attempts to differentiate by claiming superior performance, availability, functionality, or economy, they are virtually identical in their reliance on three key principles: Codd's relational model, the SQL language, and the ACID transaction model [1].

Table 2: Relational database system development

Year	Relational database system
1970	Codd's Paper
1974	System R
1978	Oracle
1980	Commercial Ingres
1981	Informix
1984	DB2
1987	Sybase
1989	Postgres
1989	SQL Server
1995	MySQL

2.1.3 The next generation database system

By the middle of the 2000s, the relational database seemed completely entrenched. In fact, the era of complete relational database supremacy was just about to end. The difference in application architectures between the client-server era and the era of massive web-scale applications created pressures on the relational database that could not be relieved through incremental innovation [1]. Table 3 illustrates the development of today and future databases. In section 3, two promising sub-areas of future databases will be introduced in detail.

Table 3: Today and future database development

Year	Today and future databases
2003	MarkLogic
2004	MapReduce
2005	Hadoop
2005	Vertica
2007	Dynamo
2008	Cassandra
2008	Hbase
2008	NuoDB

2009	MongoDB
2010	VoltDB
2010	Hana
2011	Riak
2012	Areospike
2014	Splice Machine

2.2 Summary: three platforms corresponding to three waves of databases

The three waves of databases roughly corresponds to a three waves of computer applications. The three platforms shown in figure 2 are often referred to illustrate the database system development. The first platform was the mainframe, which was supported by pre-relational database systems. The second platform, client-server and early web applications, was supported by relational databases. The third platform is characterized by applications that involve cloud computing, mobile presence, social networking, and the Internet of Things. The third platform demands a third wave of database technologies that include but are not limited to relational systems [1]. Figure 2 summarizes how the three platforms correspond to the three waves of database revolutions.

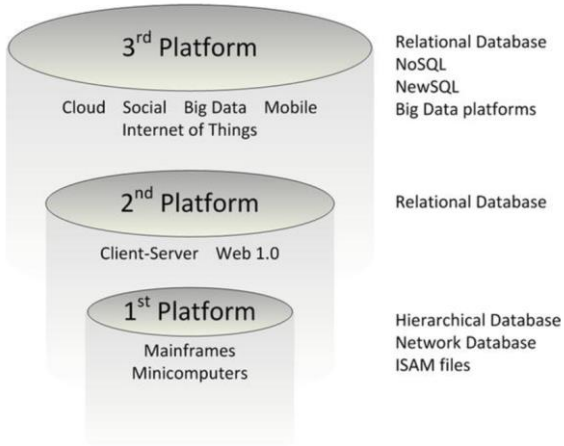

Figure 2. Three platforms correspond to three waves of database technology [1].

3. TWO PROMISING FUTURE DATABASES

The relational database was already well established. However, driven by the demands of modern applications that require global scope and continuous availability, relational databases were inadequate to deal with the volumes and velocity of the big data. In particular, the difference in application architectures between the client-server era and the era of massive web-scale applications created pressures on the relational database that could not be relieved through incremental innovation. Scalability challenges exist in scaling their infrastructure from thousands to millions of users. Even the most expensive commercial Relational Database Management System (RDBMS) such as Oracle could not provide sufficient scalability to meet the demands of these sites. Sharding at sites like Facebook has allowed a MySQL-based system to scale up to massive levels. However, there are downsides to doing this because many relational operations and database-level ACID transactions are lost which becomes impossible to perform joins or maintain transactional integrity across shards [1].

Finally, the operational costs of sharding, together with the loss of relational features, made many seek alternatives to the Relational Database Management System (RDBMS) [1].

3.1 NoSQL database

A NoSQL (**N**ot **O**nly **SQL**) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. NoSQL databases operate without a schema, allowing you to freely add fields to database records without having to define any changes in structure first. This is particularly useful when dealing with non-uniform data and custom fields. In summary, the common characteristics of NoSQL databases are:

(1) The do not use the relational model;
(2) The run well on clusters;
(3) Usually are open-source;
(4) They're built for the 21st century web estates;
(5) They're for the most part, schemaless;
(6) The most important result of the rise of NoSQL is Polyglot Persistence.

There are commonly 4 main types of NoSQL data models: key-value databases; document databases; column databases and graph databases.

3.1.1 Key-Value databases

A key-value database, or key-value store, is a data storage paradigm designed for storing, retrieving, and managing associative arrays which is a data structure more commonly known today as a dictionary or hash. In the following scenarios, it is beneficial to apply key-value databases [4].

(1) Storing Session Information;

Generally, every web session is unique and is assigned a unique SessionID value. Applications that store the SessionID on disk or in a RDBMS will greatly benefit from moving to a key-value store, since everything about the session can be stored by a single PUT request or retrieved using GET. This single-request operation makes it very fast, as everything about the session is stored in a single object. Solutions such as Memcached are used by many web applications, and Riak can be used when availability is important [1] .

(2) User Profiles, Preferences;

Almost every user has a unique UserId, Username, or some other attribute, as well as preferences such as language, color, time-zone and so on. This can all be put into an object, so getting preferences of a user takes a single GET operation. Product profiles can be stored, similarly.

(3)Shopping Cart Data;

E-commerce websites have shopping carts tied to the user. As we want the shopping carts to be available all the time, across browsers, machines, and sessions, all the shopping information can be put into the value where the key is the userid. A Riak cluster would be best suited for these kinds of applications.

3.1.2 Document databases

A document database is designed to store semi-structured data as documents, typically in JSON or XML format. It is beneficial to use document databases in the following scenarios: event logging; content management systems, blogging platforms; web analytics or real-time analytics and e-commerce Applications.

3.1.3 Column databases

A column store database is a type of database that stores data using a column oriented model. It is beneficial to use document databases in the following scenarios: event logging; content management systems, blogging platforms; counters and expiring usage.

3.1.4 Graph databases

A graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. It is beneficial to use document databases in the following scenarios: connected data; routing, dispatch, and location-Based services; recommendation engines.

3.2 NewSQL database

The term NewSQL is not quite as broad as NoSQL. NewSQL is a term to describe a new group of databases that share much of the functionality of traditional SQL relational databases, while offering some of the benefits of NoSQL technologies. NewSQL systems offer the best of both worlds: the relational data model and ACID transactional consistency of traditional operational databases; the familiarity and interactivity of SQL; and the scalability and speed of NoSQL. Some offer stronger consistency guarantees than are available with NoSQL solutions, although others limit this to 'tunable' consistency and thus aren't fully ACID-compliant [1]. The NewSQL advantages include:

(1) Minimize application complexity, stronger consistency and often full transactional support.
(2) Familiar SQL and standard tooling.
(3) Richer analytics leveraging SQL and extensions.
(4) Many systems offer NoSQL-style clustering with more traditional data and query models.

The NewSQL disadvantages include:

(1) No NewSQL systems are as general-purpose as traditional SQL systems set out to be.
(2) In-memory architectures may be inappropriate for volumes exceeding a few terabytes.
(3) Offers only partial access to the rich tooling of traditional SQL systems.

4. DATABASE CONSIDERATIONS AND REQUIREMENTS

The first and most obvious purpose of a database is to store, update, and access data. All database systems allow these operations in one form or another. Other functional and nonfunctional system considerations and requirements for choosing appropriate database systems in different applications include: (1) consistency, availability, and partition tolerance (CAP); (2) robustness and reliability; (3) scalability; (4) performance and speed; (5) partitioning ability; (7) in-database analytics and monitoring; (8) operational and querying capabilities; (9) storage management; (10) talent pool and availability of relevant skills; (11) database integrity and constraints; (12) data model flexibility; (13) database security and so on. Figure 3 shows how to make decisions involved in choosing the correct database.

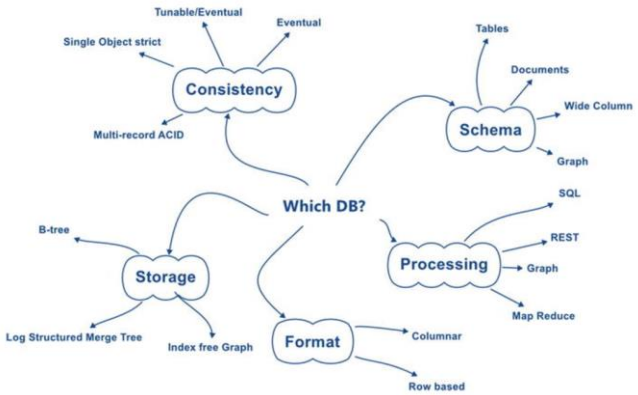

Figure 3. Decisions involved in choosing the correct database [1].

5. CONCLUSIONS

It's an exciting time to be working in the database industry. For a generation of software professionals, innovation in database technology occurred largely within the constraints of the ACID-compliant relational databases. Now that the hegemony of the RDBMS has been broken, we are free to design database systems whose only constraint is our imagination. It's well known that failure drives innovation. Some of these new database system concepts might not survive the test of time; however, there seems little chance that a single model will dominate the immediate future as completely as had the relational model. Database professionals will need to choose the most appropriate technology for their circumstances with care; in many cases, relational technology will continue be a better fit—but not always [1].

NoSQL, NewSQL, and Big Data are in many respects vaguely defined, overhyped, and overloaded terms. However, they represent the most widely understood phrases for referring to next-generation database technologies [1].

Loosely speaking, NoSQL databases reject the constraints of the relational model, including strict consistency and schemas. NewSQL databases retain many features of the relational model but amend the underlying technology in significant ways. Big Data systems are generally oriented around technologies within the Hadoop ecosystem, increasingly including Spark [1].

6. REFERENCES

[1] Harrison, Guy. Next Generation Databases. *Publisher: Apress*. December 26, 2015
[1] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems (7th Edition). *Publisher: Pearson*. June 18, 2015
[3] Hugh E.Williams, David Lane. Web Database Applications with PHP and MySQL. *Publisher: O'Reilly Media*. May 2004
[4] Haseeb, Abdul, and Geeta Pattun. "A review on NoSQL: Applications and challenges." *International Journal of Advanced Research in Computer Science 8.1* (2017).
[5] Codd, Edgar F. "A relational model of data for large shared data banks." *Communications of the ACM* 13.6 (1970): 377-387.