# Evolving databases And it's future aspects

Ajay Kumar Sahu(19111003)

January 2022

## 1 Abstract

A good database is the first step in establishing a meaningful data. It should be stored in a way that it could become information for the reader and for whoever accessing it. The growth of a database can be determined by the various challenges it faces. Doing so helps researchers develop new ideas and combinations. A data must survive for as long as it is valuable, which means it must be able to adapt in response to the changing demands of the application systems that utilise it. This might include changes in data, meta-data, programmes, and applications, as well as how users perceive the information models. Programs/application are defined by their requirement to store enormous volumes of data, the structure of which must develop as the applications that utilise it. This necessitates the data to be dynamically mapped to a growing schema.

In this paper we will be discussing various phase of DBMS in which it has been evolve to the present status of the database management system and it's scope on various domains.

## 2 Introduction

A database is a collection of data, typically describing the activities of one or more related organizations. Database management systems may be thought of as a subset of computer science in general. Languages, object-oriented programming, and other programming paradigms, compilation, operating systems, concurrent programming, data structures, algorithms, theory, parallel and distributed systems, user interfaces, expert systems and artificial intelligence,

statistical techniques, and dynamic programming are among them.

A database management system, or DBMS, is software that makes it easier to manage and use huge amounts of data. A database is a collection of information that usually describes the activity of one or more organisations. The DBMS provides various functions that allow entry, storage and retrieval of large quantities of information and provides ways to manage how that information is organized. The demand for such systems, as well as their application, is continuously increasing. Instead of employing a database management system, you may store the data in files and manage it with application-specific code. The user of a database management system (DBMS) is ultimately concerned with a real-world business, and the data to be stored defines different parts of that business. It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

When any of a table's characteristics has redundancy in values, DBMS improves data organisation by using a database schema design approach called normalisation, which separates a large table into smaller tables. DBMSs provide a number of advantages over traditional file systems, including greater flexibility and a more sophisticated backup mechanism.

## 3 Different phases of evolution in DBMS

Charles Bachman invented IDS (Integrated data store), a network database, in the early 1960s, and it impacted other DBMS for future progress. It was created to boost performance using the hardware available at the time. IBM launched the IMS (Information Management System) in 1966, which was based on a hierarchical database. It was created to store stock information for very large bills of materials for the Saturn V moon rocket and the Apollo spacecraft.

### 3.1 RDBMS

E. F. Codd introduced the relational data model in 1970. For the model, it was a theoretical paper that was presented. Following

that, E. F. Codd delivered a series of papers. According to him, data in a relational database is represented in table form. The data is accessed without the use of an algorithm, and a non-procedural high-level language is employed. The Relational Model was introduced as a standard technique for DBMS in 1980. SQL is a query language that has been accepted by ANSI and ISO. Many relational DBMS, such as Oracle, DB2, and Informix, were introduced in the same year. Researchers discovered that relational models do not accommodate multimedia data, unstructured data, or inheritance relationships after adopting RDBMS. The relational paradigm does not facilitate application scaling. When numerous tables are linked together, query processing becomes inefficient. The cost of linking tables is considerable because RDBMS join operations are costly.

### 3.2 Object-Relational Database

The object-relational database was created in 1990. The object-relational database concept combines the object notion with the relational database concept. It's like combining relational databases with OOP principles (inheritance, encapsulation, and polyphormism). This concept was introduced by Stonebaker et al., who advocated that RDBMS capabilities be expanded by supporting rich object structures. Data structures (object table for storage), Integrity constraints (object identification, relationship), and Operations are all essential components for an object relational model. Object- Relational's major purpose is to bridge the gap between relational databases and object-oriented modelling techniques used in programming languages like Java and C++. While developing applications, in order to achieve storage for complicated data types and to display relationships. A user may obtain transaction management, scalability, flexibility, performance management, user-defined data types, and much more by combining these two technologies. Increased complexity, as well as the expense that comes with it, there is a significant semantic gap between two technologies. Object applications, unlike relational-based systems, are not data-centric.

### 3.3 Object- Oriented Database management system

The OODBMS (Object- Oriented Database management system) was created around 1990. The idea behind OODBMS is to combine the principles of object-oriented programming with database administration. Encapsulation, complex objects, types and classes, extensibility, persistence, and other characteristics are supported. In an OODBMS, data is represented in the form of classes and objects rather than records, as in the E-R paradigm. Objects are run-time entities that play a critical role in real-world problems. All values in an object are kept in the form of instance variables. The term "class" refers to a group of similar objects with similar values and methods. By invoking the methods of other objects, two objects can interact with each other. Because the method and data are not visible outside the object, it provides data abstraction. It was designed to store and manage object created by programming languages like java, C++ and provide object-oriented features to users. It was designed to satisfy the demand for better model for real world entities and to provide richer data model than traditional data model.

There was a lack of flexibility with an OODBMS, as each change in the schema forces the user to alter other classes as well. It was language-dependent since the API was bound to a certain language. The queries that can be run on the data in an OODBMS are heavily reliant on the system's architecture. Other difficulties included a lack of security, a rise in complexity, a lack of standards, and a lack of experience. OODB does not support Views, despite several proposals. It was difficult to create an object-oriented view using a model that had an object identification feature.

### 3.4 Multi-Dimensional Database

In this data model, multidimensional structures are used to organise data and describe relationships between recorded data. These structures are represented as broken cubes, which are used to store data and allow users to retrieve data from them. These databases make use of an application for online analytical processing. The relational implementation of a multidimensional data model is represented by the star and snowflake schemas. The concept behind

this data model was that a relational model may be represented as a table with rows and columns, just as a multi-dimensional data base can be represented as a cube with many dimensions. When making cubes, two factors are taken into account: dimension and facts. Facts are numerical quantities regarding data such as number, size, or magnitude that assist split data into similar groups. It was created with data analysis and data storage in mind. It was created with the goal of achieving improved data extraction and a scalable data model. Updating and finding data might be difficult. It was unable to handle a database with sparse data.

### 3.5 NoSQL

Non-tabular databases (sometimes known as "not simply SQL") store data differently from relational tables. NoSQL databases are classified according to their data model. key-value, Column Oriented stores, Document, and graph are the most common kinds. They have adaptable schemas and can handle big volumes of data and heavy user loads with ease.

1. **Key-value Stores (KVS)**: The foundation of a key-value store is a key-value pair, which is akin to a dictionary. It is a well-known data model that is related to the associative array (which is built on key-value pairs). It was discovered to be extremely efficient due to the temporal complexity of its access data, which is O(1). In this architecture, each value is associated with a unique key, which is used to access and retrieve data from or into the data store. It is a schema-free paradigm that allows for quick processing. In this model, only queries using keys are allowed since values are stored with a specific key.

2. **Column Oriented Stores:** Column oriented stores are built on tables and rows, but they differ from typical databases in a number of ways. A number of columns related to a certain row can be added, much like in a column-oriented database. That row has a unique identifier that serves as the table's primary key. A table can have a number of rows with unique identifiers. There might be millions of columns in each row. The term

"column family" refers to the grouping of these columns. It also lacks a schema, although its core foundation is similar to that of a relational database. These things are from the Column shop. This store has the ability to handle a big volume of data.

3. **Document oriented Stores:** Based on the name, many people assume this store is for document storage or a document management system. However, the term "document-oriented" refers to a flexible grouping of Key/Value pairs. However, instead of separating the document to store in a key-value store, the data is saved as a single document. JSON is used to store documents in document oriented storage. A document can have a variety of structures, including complicated structures (nested objects), and no specific schema is required.

4. **Graph Database**: Graph theory is the notion that led to the creation of graph databases. Graph theory is a mathematical concept that uses nodes, edges, and vertices to express object relationships. Similar to a graph database, which stores relationships between nodes and is specialised in managing densely linked nodes and efficient in traversing relationships between distinct items. It also provides graphic algorithm and allow graphic query in addition to storing nodes, edges, and edge weights.

It was created to address the issue of scalability (the Big Data Challenge) and to improve availability. There is no reliance on any certain schema, and there is a lack of complexity (No Join Operations). It was created with the goal of achieving consistency in a distributed system. The CAP-Theorem states that any attribute between consistency and availability will be sacrificed during partition tolerance. It lacks experience and does not allow firm ACID transactions like a relational database.

### 3.6 NewSQL

In the database world, NewSQL is also a new phrase. NewSQL's core goal is to deliver scalability, availability, and robust consistency without compromising transaction capabilities. Conceived in 2011 to address challenges faced by traditional SQL-based systems,

NewSQL was designed for online transaction processing (OLTP) systems, while complying with atomicity, consistency, isolation and durability (ACID). NewSQL architecture natively supports applications that have a large number of transactions, are repetitive in their processes and utilize a small subset of data retrieving processes.

# 4 Emerging Future trends in DBMS

The need for database servers is surging as the number of users in our digital world grows at a breakneck pace. Servers may now run at lightning speed because to newer technologies and concepts such as hardware upgrades in RAM and CPU, indexing, and so on. These latest technologies have made it feasible to retrieve thousands of records in a couple of seconds and manage large amounts of data on servers with great security.

Handling and maintaining a vast amount of data that is being exchanged every day is nothing short of an uphill fight. Managing all of this data has gotten easier because to the cloud's sophisticated functionalities and computing capabilities. However, this is only the tip of the iceberg; there are other additional options for data storage that are even more efficient and effective.

## 4.1 Bridging the gap between NoSQL and SQL

SQL was initially created by IBM researchers in the mid-1970s. Carlo Strozzi coined the term "NoSQL" in 1998, which may refer to either a "No SQL" or a "Not only SQL" system. Not only are the features and functions diverse, but each database type has its own distinct personality. Basically, when SQL databases fail, NoSQL databases attempt to fill the void. In terms of function and application, these databases, whether SQL or NoSQL, offer extremely unique advantages for enterprises. Database products that don't just support a single database schema are the newest trends. It's now all about databases that combine SQL and NoSQL.

As a result of these new storage options, developers are looking at other options, and NoSQL solutions are becoming more widespread. These data storage options are now supported by frameworks and programming languages, and the use of NoSQL databases is on the rise. Partly because NoSQL systems are better suited for storing certain sorts of data, but partly because NoSQL has become a buzzword, and developers want to be a part of it.

One of the recent solution is NewSQL which is a family of modern relational DBMSs that aims to deliver the same scalable performance as NoSQL for OLTP read-write workloads while providing ACID guarantees for transactions. In other words, these systems aim for the scalability of NoSQL DBMSs from the 2000s while maintaining the relational paradigm (with SQL) and transaction support of historical DBMSs from the 1970s and 1980s. This allows applications to run a high number of concurrent transactions in order to ingest new data and change the database's state using SQL. Developers do not have to build logic to cope with ultimately consistent updates if an application utilises a NewSQL DBMS, as they would with a NoSQL system.