SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V10I6P106

Volume 10 Issue 6, 43-53, June 2023

Original Article

Demystifying Databases: Exploring their Use Cases

Pankaj Gupta[1], Prakashkumar Patel[2]

[1]*Principal Data Engineer, Discover Financial Services, USA.*
[2]*Salesforce Inc, Carmel, USA*

Received: 29 April 2023 Revised: 04 June 2023 Accepted: 16 June 2023 Published: 30 June 2023

Abstract *- This article provides a comprehensive overview of various types of available databases and their corresponding use cases. The primary objective of publishing this paper is to examine the different types of databases that exist, the reasons behind their development, and the specific use cases they serve. Databases play a critical role in facilitating the efficient organization and effective management of data in a wide range of applications. The article begins by highlighting the significance of databases in modern data-driven environments and their essential role in ensuring effective data organization and management. It emphasizes the need for a deep understanding of different database types' unique characteristics and intended purposes to address specific requirements effectively. Therefore, it is essential to have a deep understanding of the unique characteristics and intended purposes of different database types to make well-informed decisions during the process of designing and implementing database solutions.*

Keywords *- Bigdata, RDBMS, NoSQL, SaaS, SQL, Cloud, Graph, Blockchain, Time series.*

1. Introduction

Databases have long been essential components of organizations, providing substantial benefits since their inception. Their proper implementation, tailored to specific use cases for online transaction processing (OLTP) and online analytical processing (OLAP), has fuelled remarkable growth for numerous enterprises. However, in the face of rapidly evolving technologies and the digital transformation era, traditional relational database management systems (RDBMS) are encountering challenges, particularly in accommodating new use cases involving unstructured data.

To overcome these challenges and meet evolving needs, a multitude of databases have emerged, each designed to address specific requirements and leverage technological advancements. This article aims to explore the diverse landscape of these databases, delving into their distinctive characteristics, capabilities, and applications. By gaining a comprehensive understanding of the available options, organizations can make informed decisions when selecting databases that align with their specific use cases and effectively handle the demands of modern data management.

In this article, we will examine various types of databases, including both traditional RDBMS and newer alternatives such as NoSQL, graph, time-series, columnar, and document databases. Each of these database types possesses unique features and strengths that make them suitable for specific use cases. By exploring the key attributes and intended purposes of these databases, readers will gain insights into which options best align with their data management needs.

By shedding light on the diverse database types available and their corresponding use cases, this article aims to equip readers with the knowledge necessary to make informed decisions when selecting and implementing database solutions. In an era where data is abundant, and its effective management is crucial for organizational success, staying abreast of the evolving database landscape is paramount. By embracing the right database technologies and architectures, organizations can unlock the full potential of their data assets and thrive in the digital age.

2. What is a Database?

[1] Data is a collection of a distinct small unit of information. The word 'Data' originated from the word 'datum', which means 'single piece of information.' It is the plural of the word datum. It can be used in a variety of forms like text, numbers, media, bytes, etc. It can be stored in pieces of paper or electronic memory.

A database is an organized data collection that can be easily accessed and managed [2]. The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data. You can organize data into tables, rows, and columns and index it to make it easier to find relevant information.

2.1. Evolution of Databases

2.1.1. File-Based

File-based databases were first introduced around 1968. In this type of database, data was stored and managed within a flat file structure. One notable advantage of file-based databases was the availability of different access methods, including sequential, indexed, and random access. These access methods provided flexibility in retrieving and manipulating data based on specific requirements. However, working with file-based databases required extensive programming using third-generation languages like COBOL or BASIC. Developers had to write complex code to handle data storage, retrieval, and maintenance tasks. Despite their limitations and the need for manual programming, file-based databases played a significant role in early data management systems.

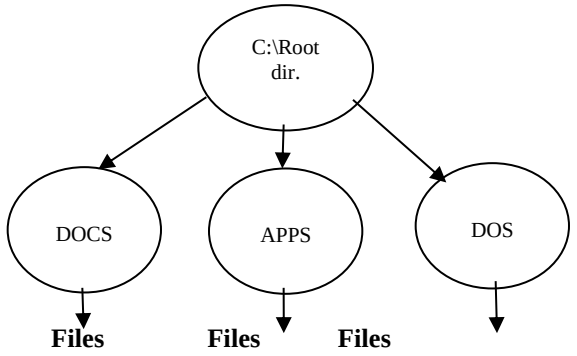


Fig 1. File-based Databases.

2.2. Hierarchical Data Model

The era between 1968 and 1980 witnessed the dominance of the Hierarchical Database model. During this period, one of the prominent hierarchical database systems was the first DBMS introduced by IBM, known as IMS (Information Management System). This model organized data in a tree-like structure, with parent-child relationships between data entities. The hierarchical database model provided efficient navigation and data retrieval, especially for applications requiring strict hierarchical relationships.[4]

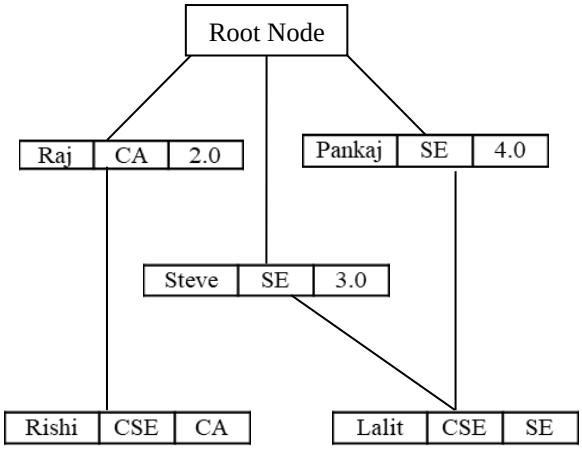


Fig. 2 Hierarchical data model

In the above-given figure, we have few students and few courses enrol, and a course can be assigned to a single student only. However, a student can enrol in any number of courses and with this, the relationship becomes one-to-many. We can represent the given hierarchical model like the below relational tables**:**

Faculty Table

Name	Dept	Course Taught
Rishi	CSE	CA
Lalit	CSE	SE

Student Table

Name	Course Enrol	Grade
Raj	CA	2.0
Steve	SE	3.0
Pankaj	SE	4.0

Fig. 3 Explanation of hierarchical model

2.3. Relational Data Model

[4] The relational data model organizes data into tables consisting of rows and columns. Each table represents a specific entity or concept, and the columns define the attributes or characteristics of the entity. Relationships between tables are established through keys, facilitating the retrieval and manipulation of data across multiple tables.

One of the key advantages of the relational data model is its ability to ensure data integrity and consistency. Through the use of constraints, such as primary keys, foreign keys, and referential integrity, the model enforces rules that maintain the integrity of the data stored within the tables.

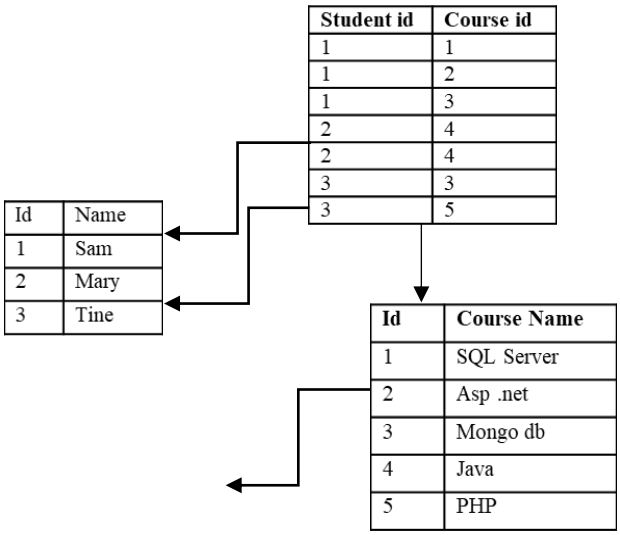


Fig. 4 Representing relational data model

In addition, the relational model provides a standardized query language known as SQL (Structured Query Language), which allows users to retrieve, manipulate, and analyse data

from a relational database using a set of declarative commands.

The simplicity and power of the relational model make it a preferred choice for organizations of all sizes and types, serving a wide range of information needs. Relational databases are utilized for tracking inventories, processing e-commerce transactions, managing vast amounts of mission-critical customer information, and much more. A relational database can be considered for any scenario where data points are interconnected and require secure, rule-based, and consistent management.

Relational databases have existed since the 1970s, and their advantages have ensured their continued dominance as the most widely accepted model for databases. While advancements and other types of databases have emerged over time, the relational model's significance remains unparalleled. The following sections will delve into these other database types and discuss their respective contributions and use cases.

3. Types of databases and their use cases

3.1. Relational Databases

[9] Relational databases are designed to organize data using tables. These tables are structured containers that enforce a specific schema onto their stored records. Each column in a table is defined with a name and a data type, specifying the kind of data it can hold. On the other hand, each row in a table represents a distinct record or data item containing values corresponding to each column.

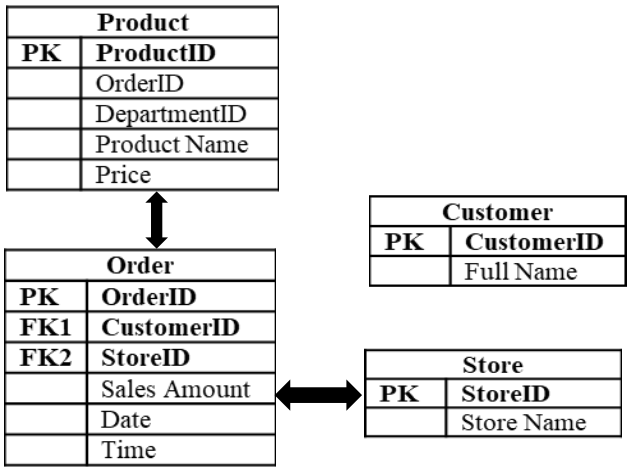


Fig. 5 Representing relational database relationship

RDBMS (Relational Database Management Systems) find extensive use in enterprise applications, e-commerce platforms, financial systems, content management systems (CMS), data warehousing, business intelligence (BI), online transaction processing (OLTP), government and public sector, healthcare systems, and more.

3.1.1 Benefits of Relational Databases

[10] They offer structured data management, efficient querying, data integrity, and scalability. RDBMS are employed for storing and managing customer data, orders, inventory, financial transactions, content, consolidated data, citizen records, medical data, and various administrative functions. Their flexibility, reliability, and support for complex querying and reporting make them indispensable in numerous industries where organized data handling and high-performance data operations are critical.

Efficient Querying

Relational databases have a special language called SQL that makes it easy for people to ask questions about the data they have stored. With SQL, you can ask for specific information, search for things that match certain criteria, and combine information from different tables. It's like having a powerful tool to find the exact data you need.

Wide Adoption and Ecosystem

Relational databases have been around for a long time and are used by many people. This means there are lots of tools, software, and resources available to help developers work with these databases. It is like having a big toolbox full of tools that make it easier to build and manage your database.

Data Relationships and Joins

Relational databases excel at handling complex data relationships. With the ability to define relationships between tables using primary and foreign keys, databases can perform efficiently joins to retrieve data from multiple tables, enabling comprehensive data analysis and reporting.

RDBMS (Relational Database Management Systems) are particularly well-suited for Online Transaction Processing (OLTP), E-commerce Platforms, and Enterprise Applications. However, they are also widely applicable in many other scenarios where there is a need for organized data management, reliable data integrity, complex querying, and scalability. The flexibility and maturity of the relational database ecosystem make them a dependable choice for various industries and use cases.

3.2. Object-Oriented Databases

[11] An object-oriented database is a type of database that is based on the principles of object-oriented programming (OOP). In an object-oriented database, data is organized and stored as objects, which are self-contained units that contain both data and the operations or methods that can be performed on that data. This allows for efficiently representing and managing complex data structures and relationships.

The object-oriented data model is a developed data model. This model can store audio, video, and graphics files. These consist of a data piece and the methods, which are the

DBMS instructions. There are two kinds of object-oriented databases, as follows:

A multimedia database stores media, such as images, that a relational database cannot store.

A hypertext database allows for linking any object to any other object. It helps to organize disparate data.

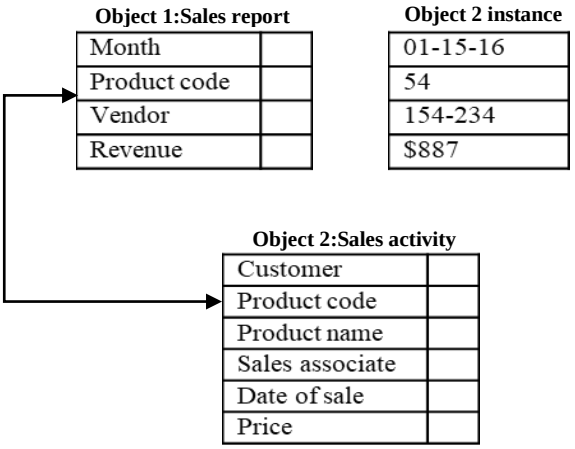


Fig. 6 Representing an object-oriented database model

3.2.1 Benefits of Object-Oriented Databases

Native Object Representation

[12] Object-oriented databases store and retrieve objects directly, preserving their inherent structure, relationships, and behaviour. This eliminates the requirement to map objects to relational tables, enhancing performance and decreasing complexity.

Complex Data Modelling

Object-oriented databases support complex data models, allowing the representation of intricate relationships, inheritance hierarchies, and complex data structures. This flexibility enables developers to model real-world scenarios more accurately and intuitively.

Persistence of Object Graphs

Object-oriented databases persist in entire object graphs, including the relationships between objects. This means that objects can be saved and retrieved as complete entities, preserving their state and relationships over time, particularly useful for complex data scenarios.

Object-oriented databases are often used in applications that require the efficient management of complex data structures and relationships, such as CAD/CAM systems, geographic information systems, and document management systems. They are also well suited for applications that require integrating different data types and sources, such as multimedia data or data from multiple sources. However, object-oriented databases can be more difficult to learn and use compared to other database models and may require specialized expertise to set up and manage.

A few examples of object-oriented databases are:

1. MongoDB
2. Apache Cassandra
3. Object DB
4. ZODB (Zope Object Database)

3.3. NoSQL and Big Data Databases

[13] NoSQL, also referred to as "not only SQL" or "non-SQL", is an approach to database design that enables storing and querying data outside the traditional structures found in relational databases. While it can still store data found within relational database management systems (RDBMS), it just stores it differently than RDBMS. The decision to use a relational database versus a non-relational database is largely contextual, and it varies depending on the use case.

Instead of the typical tabular structure of a relational database, NoSQL databases house data within one data structure, such as a JSON document. Since this non-relational database design does not require a schema, it offers rapid scalability to manage large, typically unstructured data sets.

NoSQL is also a distributed database, meaning that information is copied and stored on various servers, which can be remote or local. This ensures the availability and reliability of data. If some of the data goes offline, the rest of the database can continue to run.

NoSQL provides other options for organizing data in many ways. By offering diverse data structures, NoSQL can be applied to data analytics, managing big data, social networks, and mobile app development.

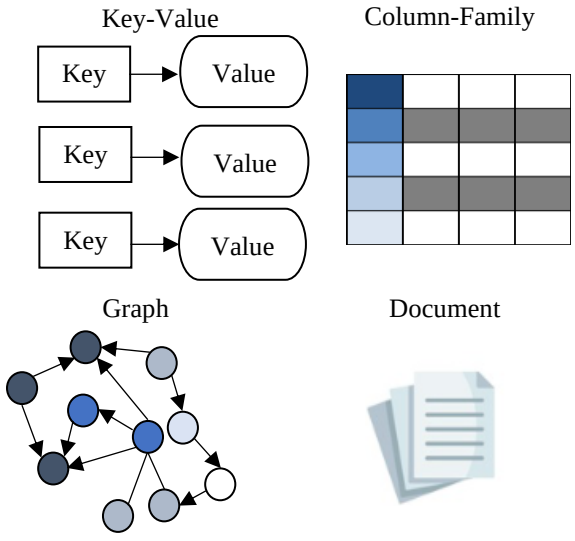


Fig. 7 Representing NoSQL databases

3.3.1. Benefits of No SQL Databases

Flexibility in Data Models

[14] NoSQL databases offer flexible data models that handle unstructured, semi-structured, and evolving data. They allow for schema-less or schema-flexible designs, enabling developers to adapt to changing data requirements quickly.

Big Data and Unstructured Data Support

NoSQL databases are well-suited for handling big data and unstructured data types, such as text, documents, graphs, and time-series data. They provide efficient storage and retrieval mechanisms for these data formats.

Distributed Computing

NoSQL databases often have built-in support for distributed computing frameworks, allowing them to integrate with technologies like Apache Hadoop or Apache Spark. This enables large-scale data processing and analytics across distributed environments.

NoSQL databases are well-suited for various scenarios where there is a need to manage large amounts of constantly changing data that does not have a fixed structure. They are commonly used in applications like social media platforms, content management systems, real-time analytics, Internet of Things (IoT) applications, and distributed systems. NoSQL databases provide enhanced flexibility, scalability, and performance compared to traditional relational databases. This means that developers can efficiently build and scale applications that require effective handling of diverse and evolving data sources.

3.4. Graph Database

Graph databases are purpose-built to store and navigate relationships. Relationships are first-class citizens in graph databases, and most of the value of graph databases is derived from these relationships. Graph databases use nodes to store data entities and edges to store relationships between entities. An edge always has a start node, end node, type, and direction; an edge can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have.

[15] A graph in a graph database can be traversed along specific edge types or across the entire graph. In graph databases, traversing the joins or relationships is very fast because the relationships between nodes are not calculated at query times but are persisted in the database. Graph databases have advantages for use cases such as social networking, recommendation engines, and fraud detection when you need to create relationships between data and quickly query these relationships.

The following graph shows an example of a social network graph. Given the people (nodes) and their relationships (edges), you can find out who the "friends of friends" of a particular person are—for example, the friends of Howard's friends.

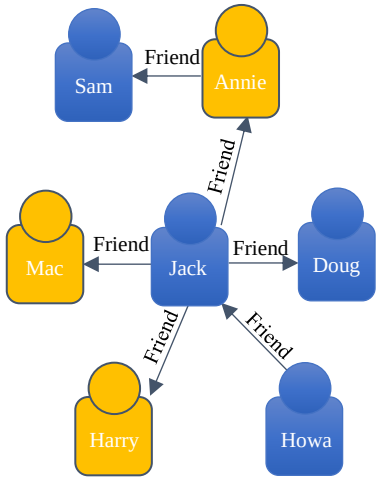


Fig. 8 Representing a graph database model

Graph databases are designed to handle and represent complex relationships between data elements. They excel at managing interconnected data and are used in various use cases.

3.4.1. Social Networks

Graph databases are well-suited for social networking platforms, where relationships between users, friendships, followers, and interactions must be efficiently stored and traversed. Graph databases make identifying connections, recommending friends, and analysing social network patterns easy.

3.4.2. Recommendation Systems

Graph databases power recommendation engines by modelling user preferences, item relationships, and user-item interactions. They allow efficient querying and traversing of the graph to provide personalized recommendations for products, movies, music, articles, and more.

3.4.3. Fraud Detection

Graph databases are used for fraud detection systems, where connections and patterns among users, transactions, and behaviours need to be analysed in real-time. Graph databases can identify suspicious patterns, detect fraud networks, and provide insights into fraudulent activities.

3.4.4. Knowledge Graphs

Graph databases are employed in knowledge management systems, where structured and unstructured data from various sources are linked together to create a comprehensive knowledge graph. This enables semantic

search, intelligent recommendations, and the discovery of relationships among concepts, entities, and information.

3.4.1 Benefits of Graph Databases
Relationship-Centric Data Model

Graph databases are designed to store and represent data in terms of nodes (entities) and edges (relationships) between those nodes. This relationship-centric model allows for efficient storage and querying of complex relationships and interconnected data.

Efficient Relationship Navigation

Graph databases excel at traversing relationships between nodes. With their native ability to navigate connections, they can quickly retrieve related data, making queries that involve complex relationships more efficient and performant.

Data Integrity and Consistency

Graph databases enforce referential integrity and consistency through relationships. Changes made to nodes or relationships are immediately reflected throughout the graph, ensuring data integrity and eliminating the need for complex join operations.

3.5 Time Series Databases

[16] A time series database (TSDB) is a database optimized for time-stamped or time series data. Time series data are simply measurements or events that are tracked, monitored, down-sampled, and aggregated over time. This could be server metrics, application performance monitoring, network data, sensor data, events, clicks, trades in a market, and many other types of analytics data.

A time series database is built specifically for handling time-stamped metrics and events or measurements. A TSDB is optimized for measuring change over time. Properties that make time series data very different from other data workloads are data lifecycle management, summarization, and large-range scans of many records.

In most definitions, time-series data, often referred to as time-stamped data, is a sequence of values indexed in time order. Time stamping refers to data collected at various times, where each value is time stamped. These data points are usually gathered from the same source and are used to measure progress over time.

You can use relational or NoSQL databases to crunch time-series data, but purpose-built time-series databases are tailored to exploit the unique features of time-series data. This implies that time-series databases ingest at a faster rate, query more quickly and compress data more efficiently. Furthermore, time-series databases include special analytical capabilities and management features not found in most relational or NoSQL databases.

This list is not exhaustive, but here are some advantages you might get from using a time-series database.

- Time-series databases can handle high-velocity data very well.
- Time-series databases are purpose-built for storing time-series data, making them more efficient in storage and querying.
- Time-series databases often have built-in analytics and management features designed specifically for time-series data.
- Many time-series databases are open source, which means they’re free to use.
- There are also some disadvantages that you should be aware of before using a time-series database:
- Time-series databases are often more complex to set up and manage than relational or NoSQL databases.
- There are relatively few time-series databases to choose from, so you might not have as much flexibility in choosing a platform that meets your specific needs.
- Time-series data can be very large, so you’ll need enough storage capacity to accommodate your data.
- Use cases for time-series databases:
- Monitoring software systems and bare-metal hardware systems.
- Continually capturing metrics from Internet of Things (IoT) devices.
- Financial trading systems.
- Recording stock prices over time.
- Asset-tracking applications.

A few examples of time series databases are: -

- InfluxDB.
- Kdb+
- Prometheus.
- Graphite.
- TimescaleDB.
- DolphinDB.
- RRDTool.
- OpenTSDB.

3.6. Blockchain Databases

[17] A blockchain database is like a digital ledger shared across many computers. Each block in the chain holds a bunch of transactions, with a special code proving the transactions are real. Once a block is added to the chain, it can't be changed or removed, which keeps the data safe and reliable.

What's special about a blockchain database is that it doesn't have a central authority controlling it. Instead, it is spread out among different computers, and everyone who has a copy gets to help decide how it works.

People love blockchain because it's secure, transparent, and can't be messed with easily. It's used for things like digital money, keeping track of products in a supply chain, and even voting systems. It's a cool way to store and transfer data without having to trust just one person or company.

3.6.1. Benefits of Blockchain Database

Decentralized Control

In general, blockchains enable multiple parties to share information without the need for a central administrator. As mentioned earlier, the consensus mechanism plays a crucial role in decision-making within blockchains. However, databases serve a completely different purpose. Databases typically require a central administration because there are situations where consensus may not be reliable. In certain cases, the expertise or judgment of a single individual may prove to be more effective than the collective wisdom of a large group.

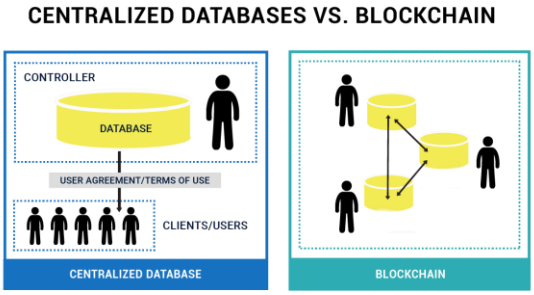


Fig. 9 Representing a difference between centralised vs blockchain database

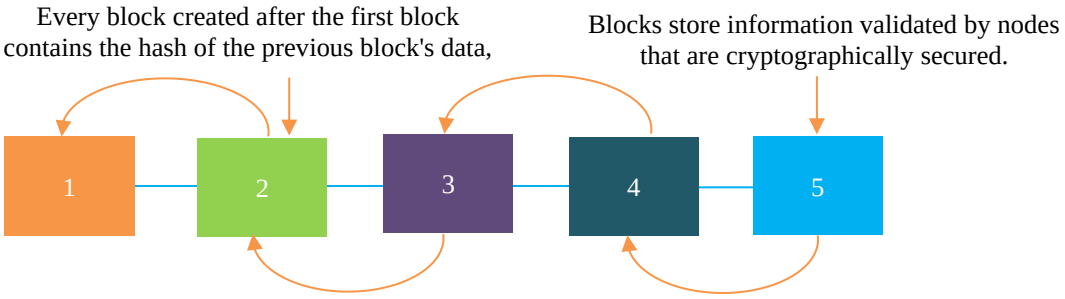


Fig.10 Block creation in a blockchain database

History of Itself

Centralized databases are designed to store and retrieve current information without maintaining a record of past transactions. However, blockchains operate differently. They not only store real-time information but also have the capability to trace and retain the history of previous transactions. Blockchains create databases that act as growing archives, maintaining a comprehensive and unalterable record of their transaction history. This feature allows for transparency, accountability, and the ability to verify and audit past activities within the blockchain network. As a result, blockchains provide a robust and reliable means of storing and accessing information while preserving a complete and traceable transaction history.

Performance

[7] Blockchains, although commonly used as reliable systems for recording transactions and serving as transaction platforms, are perceived to have a similar level of speed as traditional databases when it comes to digital transaction technology. Undoubtedly, improvements in the performance and nature of blockchain technology are anticipated, but databases have also made significant strides in performance over the course of several decades.

In terms of confidentiality, a permissioned blockchain, similar to a centralized database, can exert control over both write and read access. However, if confidentiality alone is the primary objective, blockchains do not possess a distinct advantage over centralized databases. The choice between the two ultimately depends on the specific requirements and objectives of the system in question.

Blockchain databases can be used in various applications like Cryptocurrencies, voting Systems, decentralized applications, Internet of Things.

A few blockchain-based databases are:

- BigchainDB
- Apache Cassandra
- ChainifyDB
- CovenantSQL
- Modex BCDB
- Postchain
- ProvenDB

3.7. Cloud Databases

[8] A cloud database is a database that is deployed, delivered, and accessed in the cloud. Cloud databases

organize and store structured, unstructured, and semi-structured data like traditional on-premises databases. However, they also provide many of the same benefits of cloud computing, including speed, scalability, agility, and reduced costs.

Cloud databases, like their traditional on-premises counterparts, can be categorized into two main types:

Relational databases and non-relational databases. Relational cloud databases are structured databases consisting of tables composed of columns and rows. They enable the organization of data based on predefined relationships, helping users understand the logical connections between data points. These databases typically employ a fixed data schema, and data manipulation and retrieval can be performed using a structured query language (SQL). Relational databases are known for their high consistency, reliability, and suitability for handling large volumes of structured data. Popular examples of relational databases used in the cloud include SQL Server, Oracle, MySQL, PostgreSQL, Spanner, and Cloud SQL.

Non-relational cloud databases, on the other hand, are designed to store and manage unstructured data. This includes various data types such as text from emails and mobile messages, documents, surveys, rich media files, and sensor data. Unlike relational databases, non-relational databases do not adhere to a rigid schema structure and allow for the storage and organization of information in any format, irrespective of its structure. Examples of non-relational databases commonly used in the cloud are MongoDB, Redis, Cassandra, Hbase, and Cloud Bigtable.

3.7.1 Benefits of Cloud Database

Reduced Operational Overhead

Cloud databases eliminate the management and maintenance of any physical infrastructure. Your cloud provider is responsible for provisioning, updating, and maintaining all the hardware, operating systems, and database software.

Improved Agility and Scalability

You can launch a new cloud database or decommission one in minutes. This allows you to test, operationalize, and validate new ideas faster. Plus, cloud databases can dynamically scale as your applications grow and deliver consistent performance under high load.

Lower Total Cost of Ownership (TCO)

The cloud service provider owns and operates infrastructure allowing teams to focus on building applications. In addition, pay-as-you-go options lets you provision what you need, when you need it, and scale up or down depending on your usage.

Flexible Database Options

When selecting a cloud database, you have the option to choose purpose-built databases that are specifically designed to meet the unique requirements and performance demands of your use case and applications.

Better Reliability

Cloud platforms, including cloud databases, come with a host of built-in features designed to maintain constant connectivity and fulfill SLAs, including high availability, automated backups, and robust disaster recovery.

Below are a few popular cloud databases:

- Snowflake
- Amazon Web Services (AWS) Relational Database Service (RDS)
- Microsoft Azure SQL Database
- Google Cloud SQL
- MongoDB Atlas
- Oracle Autonomous Database

Cloud databases are well-suited for Data Analytics and Business Intelligence, Web and Mobile Applications, E-commerce and Retail.

3.8. Document Databases

[5] Document databases store data in structured documents, typically represented in formats such as Extensible Markup Language (XML) or JSON. In many ways, document databases can be seen as an evolution of key-value stores. They share similarities with key-value stores, allowing for nested key-value pairs, but document databases offer improved query performance compared to traditional key-value stores.

Document databases are considered advancements over schema-less key-value stores because they introduce a self-described document format. This format enables the databases to perform more validations and checks, offering increased flexibility and reduced restrictions compared to relational database management systems (RDBMSs) with rigid schemas.

Below is an example of a sample of Sammy's contact card document.[6]

```
{
  "_id": "sammyshark",
  "firstName": "Sammy",
  "lastName": "Shark",
  "email": "sammy.shark@digitalocean.com",
  "department": "Finance"
}
```

The following is another sample document representing a colleague of Sammy's named Tom, who works in multiple departments and also uses a middle name:

```
{
    "_id": "tomjohnson",
    "firstName": "Tom",
    "middleName": "William",
    "lastName": "Johnson",
    "email": "tom.johnson@digitalocean.com",
    "department": ["Finance", "Accounting"]
}
```

This second document has a few differences from the first example. For instance, it adds a new field called middleName. Also, this document's department field stores not a single value but an array of two values: "Finance" and "Accounting".

Documents in document databases contain various data fields, implying that they possess distinct schemas. In the context of a database, a schema refers to its formal structure, specifying the types of data it can accommodate. In the case of documents, their schemas are manifested through the names of their fields and the corresponding values that these fields represent.

3.8.1 Benefits of Document Databases

Flexibility and Adaptability

[6] Document databases provide a high level of control over the data structure, allowing for easy experimentation and adaptation to emerging requirements. This flexibility is due to the ability to add new fields instantly and modify existing ones at any time. Developers have the autonomy to determine whether changes should be applied retroactively to old documents or if they will only affect future data. This empowers developers to swiftly respond to evolving needs and make adjustments to the data structure as necessary.

Ability to Manage Structured and Unstructured Data

[6] Relational databases excel at storing data that adhere to a strict structure. However, document databases offer versatility by effectively handling both structured and unstructured data. While structured data can be easily represented in a tabular format, similar to a spreadsheet with rows and columns, unstructured data is more complex to categorize. Unstructured data encompasses various forms, such as social media posts containing textual content and multimedia elements, server logs with diverse formats, or data streams from diverse sensors in smart homes. Document databases provide a suitable solution for efficiently managing and organizing this unstructured data, allowing for greater flexibility and adaptability in handling diverse data types.

Scalability by Design

[6] Relational databases are often write-constrained, and increasing their performance requires you to scale vertically (meaning you must migrate their data to more powerful and performant database servers). Conversely, document databases are designed as distributed systems that instead allow you to scale horizontally (meaning that you split a single database up across multiple servers). Because documents are independent units containing both data and schema, it's relatively trivial to distribute them across server nodes. This makes it possible to store large amounts of data with less operational complexity.

Some popular document databases include:
1. MongoDB
2. Couchbase
3. Apache Cassandra
4. Amazon DocumentDB
5. Elasticsearch

Content Management Systems (CMS) benefit greatly from using document databases due to their ability to efficiently store and manage unstructured content. Document databases provide a flexible and scalable solution for handling various types of content, including articles, blog posts, and multimedia files.

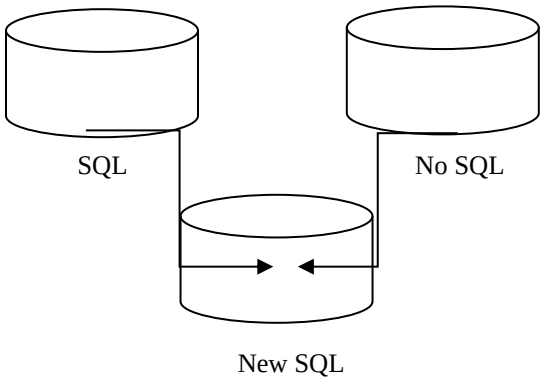
User Profiles and Personalization benefit greatly from the use of document databases. Document databases provide a flexible and efficient solution for storing user profiles, including diverse data fields, preferences, and personalized settings.

3.9. NewSQL and Hybrid Databases

NewSQL databases are modern SQL databases that address key challenges associated with traditional online transaction processing (OLTP) relational database management systems (RDBMS). They aim to achieve the scalability and improved performance typically associated with NoSQL databases while retaining the advantages of traditional DBMS.

In essence, NewSQL databases combine the benefits of both worlds by providing the scalability and performance of NoSQL databases while preserving the ACID (Atomicity, Consistency, Isolation, and Durability) guarantees of traditional RDBMS. ACID transactions ensure the integrity of business processes, concurrency control for multiple transactions, data durability in the event of system failures or errors, and consistency before and after a transaction.

In-memory storage and data processing are indeed key features of many NewSQL databases. NewSQL databases store data in the servers' main memory (RAM) rather than on traditional disk storage by utilising in-memory technology.



3.9.1 Benefits of NewSQL Databases

Faster Query Performance

Storing data in memory allows for significantly faster data access and retrieval. With no disk I/O delays, queries can be processed much more quickly, resulting in faster response times and improved overall query performance.

Reduced Latency

With data residing in memory, NewSQL databases can significantly reduce latency compared to disk-based systems. This is particularly beneficial for use cases that require low-latency access, such as real-time data processing, online transaction processing, and interactive applications.

Polyglot Persistence

Hybrid databases offer support for polyglot persistence, enabling them to store and manage data using multiple storage technologies simultaneously. This capability allows developers to choose the most appropriate data storage solution for each specific data type, resulting in optimized performance and cost efficiency.

Here are a few NewSQL Databases:
1. CockroachDB
2. TiDB
3. NuoDB
4. MemSQL

Some famous Hybrid Databases:
1. Apache Cassandra (wide column store)
2. Apache HBase (wide column store)
3. MongoDB (document store)
4. Couchbase (document store)

4. Conclusion

The realm of databases presents a wide array of choices to meet the diverse needs of data management. It is essential to have a comprehensive understanding of the various database types and their specific applications to make informed decisions regarding the efficient storage and management of data.

Relational databases excel in situations requiring efficient, structured data management, ensuring data integrity and facilitating complex queries. These databases are widely employed in enterprise applications, e-commerce platforms, financial systems, and other contexts where the meticulous handling of organized data and high-performance operations is of utmost importance. Object-oriented databases provide a seamless way to store and retrieve objects while preserving their inherent structure and relationships. By eliminating the need to map objects to relational tables, these databases enhance performance and reduce complexity.

Graph databases excel in representing and navigating intricate relationships between data elements. They prove invaluable in applications involving social networks, recommendation systems, fraud detection, and knowledge graphs.

Blockchain databases provide a secure, transparent, and tamper-proof means of storing and transferring data. Their applications span across cryptocurrency transactions, supply chain management, voting systems, and other contexts where data integrity and transparency are paramount.

NoSQL databases are highly proficient in managing unstructured, semi-structured, and evolving data. Their flexible data models and schema-less designs enable developers to swiftly adapt to evolving data requirements. NoSQL databases are well-suited for use cases such as social media platforms, content management systems, real-time analytics, and IoT applications.

Organizations and developers can choose the best database technology for their specific requirements by knowing the advantages and suitable purposes of different database types. The important thing is to match the characteristics and abilities of the database with the needs of the application, ensuring it performs well, can handle larger workloads, and effectively manages data.

References

[1] Java point. [Online]. Available: https://www.javatpoint.com/what-is-databaseavailable [online]

[2] Mark Whitehorn and Bill Marklyn, Inside Relational Databases with Examples in Access. [Online]. Available: https://www.google.com/books/edition/Inside_Relational_Databases_with_Example/XVrqkyceJQUC?hl=en&gbpv=1&dq=who+invented+databases&printsec=frontcover

[3] Kaiti Norton, File-Based Data Management System, 1996. [Online]. Available: https://www.webopedia.com/definitions/file-management-system/
[4] Difference between Hierarchical and Relational Data Model [Online]. Available: https://www.geeksforgeeks.org/difference-between-hierarchical-and-relational-data-model/
[5] Peng Yue, and Zhenyu Tan, GIS Methods and Techniques [Online]. Available: https://www.sciencedirect.com/topics/computer-science/document-database
[6] Mateusz Papiernik, An Introduction to Document-Oriented Databases, 2021. [Online]. Available: https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-document-oriented-databases
[7] Blockchain database [Online]. Available: https://intellipaat.com/blog/tutorial/blockchain-tutorial/blockchain-database/?US
[8] What is a Cloud Database?. [Online]. Available: https://cloud.google.com/learn/what-is-a-cloud-database
[9] What is a Relational Database (RDBMS)?. [Online]. Available: https://www.oracle.com/database/what-is-a-relational-database/
[10] Nishtha Jatana et al., "A Survey and Comparison of Relational and Non-Relational Database," *International Journal of Engineering Research & Technology*, vol. 1, no. 6, pp. 1-5, 2012. [Google Scholar] [Publisher Link]
[11] AkshitaKumawat, Definition and Overview of ODBMS. [Online]. Available: https://www.geeksforgeeks.org/definition-and-overview-of-odbms/#
[12] What Are Object-Oriented Databases And Their Advantages, 2019. [Online]. Available: https://www.c-sharpcorner.com/article/what-are-object-oriented-databases-and-their-advantages2/
[13] What are NoSQL databases?, IBM [Online]. Available: https://www.ibm.com/topics/nosql-databases
[14] Advantages of NoSQL Databases, MongoDB. [Online]. Available: https://www.mongodb.com/nosql-explained/advantages
[15] Amazon [Online]. Available: https://aws.amazon.com/nosql/graph/
[16] Time Series Database (TSDB) Explained, Influxdat. [Online]. Available: https://w2.influxdata.com/time-series-database/