Database Storage Models

Independent Seminar
submitted in partial fulfillment of the requirements
of
the degree of
Master Of Technology

by
Prabhat Pushp
National Institute of Technology, Delhi
New Delhi, India, 110036
212211012@nitdelhi.ac.in

Supervisor:
Shelly Sachdeva
National Institute of Technology, Delhi
New Delhi, India, 110036
shellysachdeva@nitdelhi.ac.in

COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY DELHI
2023

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Database Storage Models

A database storage model refers to the way in which data is organized and stored in a database. There are several different database storage models, each with its own unique features and advantages.

The relational model[1] is a database storage model that is based on the concept of organizing data into tables, with each table representing a separate entity. The tables are then linked together through the use of keys, allowing for complex relationships between data elements to be easily represented and navigated. The relational model is widely used due to its simplicity and flexibility, and is well suited for handling large amounts of data.

The object-oriented model[2] is a database storage model that is based on the concept of organizing data into objects, with each object representing a separate entity. The objects are then linked together through the use of references, allowing for complex relationships between data elements to be easily represented and navigated. The object-oriented model is ideal for handling complex data structures and is well suited for handling large amounts of data.

The database storage model plays a crucial role in determining the organization and accessibility of data within a database. Different database storage models have their own unique advantages and disadvantages, and the appropriate model should be chosen based on the specific needs and requirements of the data being stored.

1.2 Applications of Database Storage Models

The database storage model is an essential component of modern information systems and has numerous applications in various industries and fields. The model provides a structured approach to storing and organizing data in a way that allows for easy retrieval, manipulation, and analysis. The following essay will discuss some of the key applications of the database storage model, including its use in business, healthcare, and scientific research.

One of the most common applications of the database storage model is in business settings. Businesses of all sizes and industries rely on databases to store and manage their operational data, including customer information, inventory levels, financial transactions, and sales records. This data is used to inform business decisions, identify trends and patterns, and measure performance. For example, a retailer might use a database to track sales data and generate reports on the most popular products and the most profitable locations.

In the healthcare industry, the database storage model is used to store and manage patient records and other medical data. This data is critical for providing high-quality care and ensuring patient safety. For example, a hospital might use a database to store patient medical histories, medication lists, and lab results. This information can be accessed by doctors and nurses to provide personalized care and avoid adverse drug interactions or other potential complications.

The database storage model is also important in the field of scientific research. Researchers often need to collect and analyze large amounts of data to test hypotheses and draw conclusions. A database can be used to store and organize this data in a way that allows for easy access and analysis. For example, a research team studying the effects of a new drug on a particular medical condition might use a database to track and analyze the results of clinical trials.

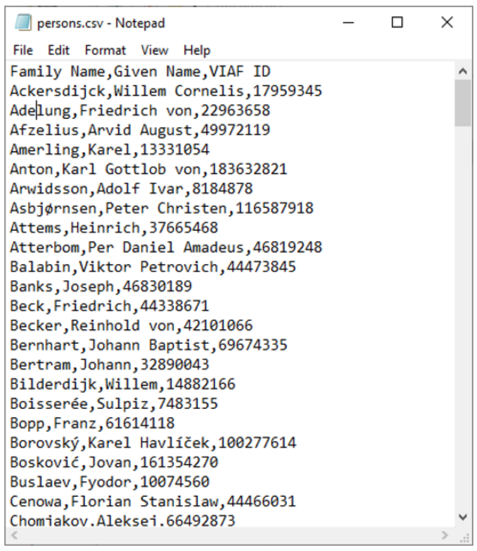
In conclusion, the database storage model is a valuable tool for storing and organizing data in a structured and efficient manner. It has a wide range of applications, including in business, healthcare, and scientific research. By providing a way to manage and analyze data, the database storage model enables organizations to make informed decisions, improve operational efficiency, and advance the understanding of complex issues.

Chapter 2

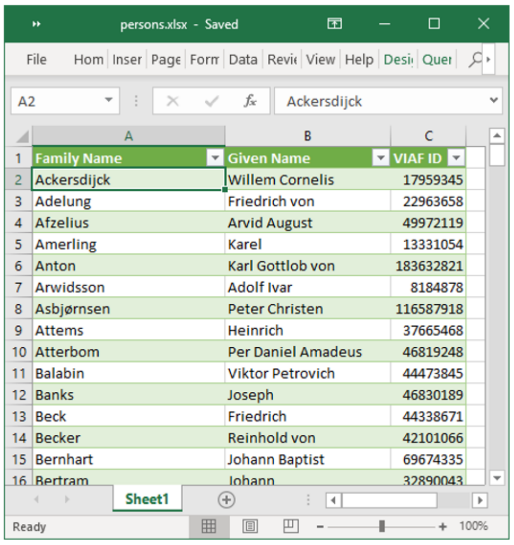
Flat File Based Databases

2.1 Introduction

A flat file based database[3] is a type of database system that stores data in a single, flat file. This file contains all the data for the database, including tables and records, and is organized in a specific format. Flat file databases are simple and easy to use, making them a popular choice for small businesses and personal use.



```
Family Name,Given Name,VIAF ID
Ackersdijck,Willem Cornelis,17959345
Adelung,Friedrich von,22963658
Afzelius,Arvid August,49972119
Amerling,Karel,13331054
Anton,Karl Gottlob von,183632821
Arwidsson,Adolf Ivar,8184878
Asbjørnsen,Peter Christen,116587918
Attems,Heinrich,37665468
Atterbom,Per Daniel Amadeus,46819248
Balabin,Viktor Petrovich,44473845
Banks,Joseph,46830189
Beck,Friedrich,44338671
Becker,Reinhold von,42101066
Bernhart,Johann Baptist,69674335
Bertram,Johann,32890043
Bilderdijk,Willem,14882166
Boisserée,Sulpiz,7483155
Bopp,Franz,61614118
Borovský,Karel Havlíček,100277614
Bošković,Jovan,161354270
Buslaev,Fyodor,10074560
Cenowa,Florian Stanislaw,44466031
Chomiakov.Aleksei.66492873
```



	A	B	C
1	Family Name	Given Name	VIAF ID
2	Ackersdijck	Willem Cornelis	17959345
3	Adelung	Friedrich von	22963658
4	Afzelius	Arvid August	49972119
5	Amerling	Karel	13331054
6	Anton	Karl Gottlob von	183632821
7	Arwidsson	Adolf Ivar	8184878
8	Asbjørnsen	Peter Christen	116587918
9	Attems	Heinrich	37665468
10	Atterbom	Per Daniel Amadeus	46819248
11	Balabin	Viktor Petrovich	44473845
12	Banks	Joseph	46830189
13	Beck	Friedrich	44338671
14	Becker	Reinhold von	42101066
15	Bernhart	Johann Baptist	69674335
16	Bertram	Johann	32890043

Figure 2.1: Flat File based databases (CSV and Excel file)[4]

However, flat file databases have some limitations compared to other types of database systems. This report will provide an overview of flat file databases, including their features, limitations, and potential uses.

2.2 Types fo Flat File Based Databases

There are many types of flat file databases. Three of them are mentioned below:

- **Text files:** These are the most common type of flat file database, and they store data in a plain text format. The data is organized into columns and rows, with each row representing a record and each column representing a field. The data is separated by a delimiter, such as a comma, tab, or semicolon, which allows for easy parsing and extraction of the data.
- **CSV (Comma-Separated Values) files:** These are similar to text files, but the data is separated by commas instead of a delimiter. This allows for easy import and export of data between different applications and systems.
- **Spreadsheets:** These are flat file databases that are commonly used in office productivity applications, such as Microsoft Excel or Google Sheets. They consist of rows and columns, and the data is organized into cells. Spreadsheets offer more functionality and flexibility than text files or CSV files, as they allow for formulas and calculations to be performed on the data.

2.3 Features of Flat File based Databases

One of the main features of flat file based databases is their simplicity. These databases consist of a single file that contains all the data for the database. This makes them easy to create and manage, as there is no need for complex database management systems or specialized knowledge.

Another key feature of flat file databases is their flexibility. These databases can be created and managed using a variety of tools and software, including text editors, spreadsheets, and specialized database software. This means that users can choose the tools that best fit their needs and preferences.

Flat file databases are also portable and can be easily transferred between different computers and operating systems. This makes them a good choice for users who need to access their data from multiple locations or devices.

2.4 Advantages and Disadvantages of Flat File based Databases

2.4.1 Advantages of Flat File Based Database

- **Easy to set up and use:** Flat file databases do not require specialized software or technical knowledge to create and manage. This makes them an accessible and user-friendly option for individuals or small businesses with limited resources.

- **Cost-effective:** Flat file databases do not require expensive hardware or software to operate, which makes them a cost-effective option for small-scale data storage. This type of database can be easily created and managed using simple text editing tools, such as Notepad or Wordpad, which are freely available on most computers.
- **Flexible data storage:** Flat file databases can store a wide range of data types, including text, numbers, images, and other multimedia files. This makes them a versatile option for data storage, as they can be used to store and manage a wide range of information.
- **Portable:** Flat file databases are stored in a single file, which makes them easy to transport and share with others. This is useful for individuals or small businesses who need to transfer data between computers or share information with others.
- **Simplicity:** Flat file databases are simple and easy to use, which makes them a good choice for individuals or small businesses who are new to data management. This type of database does not require complex queries or programming languages to access or manipulate data, which makes it a user-friendly option for data storage.

2.4.2 Disadvantages of Flat File Based Database

- **Limited scalability:** Flat file databases are not designed to handle large amounts of data, and may become slow or unresponsive when used with large datasets. This makes them a poor choice for businesses or organizations that require data storage for large numbers of users or large amounts of data.
- **Limited security:** Flat file databases are stored as a single file on a computer, which makes them vulnerable to data loss or corruption. This type of database does not have built-in security features, such as encryption or authentication, which makes it difficult to protect sensitive data from unauthorized access.
- **Limited data access:** Flat file databases are not designed for concurrent access, which means that only one user can access the database at a time. This can make it difficult for multiple users to access and manipulate data simultaneously, which can be a problem for businesses or organizations with multiple users.
- **Limited data management:** Flat file databases do not have built-in tools for data management, such as data validation or data integrity checks. This means that users must manually check and validate data to ensure that it is accurate and consistent, which can be time-consuming and error-prone.
- **Limited data analysis and reporting:** Flat file databases do not have built-in tools for data analysis or reporting, which makes it difficult to generate reports.

2.5 Applications of Flat File based Databases

Despite their limitations, flat file databases can still be useful in certain situations. For example, they may be a good choice for small businesses or personal use, where the data sets are not too large and the data structures are relatively simple.

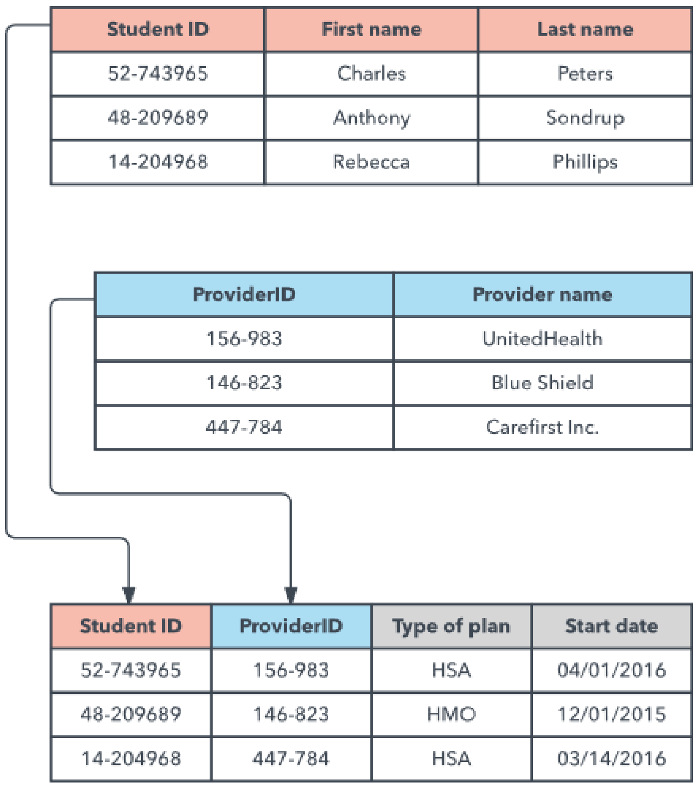
Flat file databases can also be useful for applications that require portability or flexibility. For example, a user may want to create a simple database to track their personal finances or inventory, and be able to access and update it from multiple devices. In this case, a flat file database would be a good choice because it can be easily transferred and used on different devices.

Chapter 3

Relational Based Databases

3.1 Introduction

A relational database [5] is a type of database that stores and organizes data into tables known as relations. Each table consists of columns and rows, with each column representing a specific attribute and each row representing a unique record. The data within a relational database is structured in a way that allows for easy querying and data manipulation.



Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

Figure 3.1: Relational databases[6]

The history of relational databases can be traced back to the 1970s, when IBM researcher Dr. Edgar F. Codd published a paper outlining his concept of a relational database model. Codd's paper introduced the notion of a database as a collection of tables, each table containing a set of attributes and records, and the relationships between tables defined by foreign keys.

In the years following Codd's paper, various database management systems (DBMS) were developed to implement the relational database model. These included the IBM System R, which was the first DBMS to fully implement Codd's relational model, and the Oracle database, which was developed by Larry Ellison and became one of the most widely used relational databases in the world.

The rise of the personal computer in the 1980s led to the development of smaller, more affordable DBMSs, such as Microsoft Access and FileMaker. These systems allowed individuals and small businesses to easily create and manage their own databases, greatly increasing the popularity of relational databases.

In the 1990s, the development of the internet and the proliferation of web-based applications led to the creation of new DBMSs that were specifically designed for use on the web. These included MySQL, which was developed by a team at the Swedish company T.C.X. Datakonsult AB and is now one of the most popular open-source relational databases, and Microsoft SQL Server, which is used by many large organizations and is considered one of the most powerful and scalable relational databases available.

Today, relational databases are used in a wide range of applications, from managing personal finances to running large-scale enterprise systems. The power and flexibility of relational databases make them an essential tool for organizing and accessing data in any application that requires the storage and manipulation of large amounts of information.

3.2 Types of Relational Databases

3.2.1 Centralized Relational Databases

Centralized relational databases are the most common type of relational database. In this type of database, all of the data is stored in a single location, which is typically a server. This makes it easy to manage and maintain the database, as all of the data is in one place.

One of the main advantages of centralized relational databases is that they are easy to set up and manage. They also offer good performance and reliability, as the data is stored on a dedicated server.

However, centralized relational databases can be vulnerable to data loss or corruption if the server goes down. They also require a large amount of storage space, which can be expensive.

3.2.2 Distributed Relational Databases

Distributed relational databases are a type of database that is distributed across multiple locations, such as multiple servers or computers. In this type of database, the data is replicated across the different locations, which provides several benefits.

One of the main advantages of distributed relational databases is that they are more resilient than centralized databases. If one location goes down, the data can still be accessed from another location. This makes them more reliable and reduces the risk of data loss or corruption.

Another advantage of distributed relational databases is that they can offer better performance and scalability. Since the data is distributed across multiple locations, it can be accessed from multiple points at the same time, which can improve the speed of queries.

However, distributed relational databases can be more complex to set up and manage than centralized databases. They also require more storage space, as the data needs to be replicated across multiple locations.

3.2.3 Cloud-Based Relational Databases

Cloud-based relational databases are a type of database that is hosted in the cloud, rather than on a local server. This means that the data is stored on a network of servers that are managed by a third-party provider.

One of the main advantages of cloud-based relational databases is that they are highly scalable. Since the data is stored on a network of servers, it can easily be expanded to accommodate more data as needed. This makes them suitable for use in applications that require large amounts of data storage.

3.3 Advantages and Disadvantages of Relational Databases

3.3.1 Advantages of Relational Databases

- **Data Normalization:** Relational databases follow a specific structure that allows for data to be organized and easily accessed. This structure, known as data normalization, helps to eliminate data redundancy and inconsistencies.

- **Scalability:** Relational databases are designed to handle large amounts of data, making them scalable for businesses that may experience growth or change.
- **Flexibility:** Relational databases allow for data to be easily updated, added, or removed, providing flexibility for changing business needs.
- **Security:** Relational databases offer robust security features, such as user-level access controls and data encryption, to protect sensitive information.
- **Data Integrity:** Relational databases enforce data integrity rules, such as uniqueness and referential integrity, to ensure data accuracy and consistency.

3.3.2 Disadvantages of Relational Databases

- **Complexity:** Relational databases can be complex to set up and maintain, requiring specialized knowledge and expertise.
- **Cost:** Relational databases often require the purchase of specialized software and hardware, which can be expensive for smaller businesses.
- **Performance:** Relational databases may experience performance issues when handling large amounts of data, requiring additional resources to maintain optimal performance.
- **Limited Functionality:** Relational databases are designed for structured data and may not be suitable for unstructured or complex data types.
- **Data Silos:** Relational databases can create data silos, where data is stored in separate tables, making it difficult to access and analyze data across departments or business units.

3.4 Applications of Relational Databases

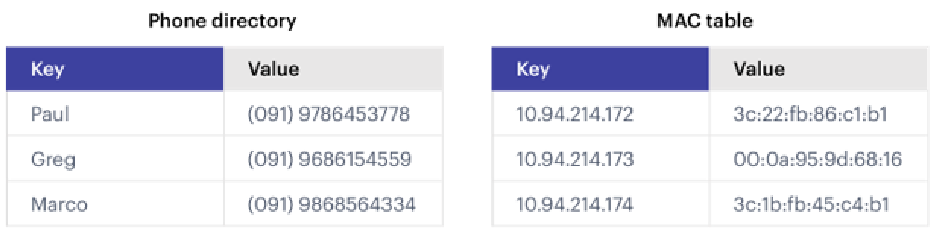
- **Data storage:** Relational databases are widely used for storing structured data, such as customer information, product inventory, and financial transactions.
- **Data analysis:** Relational databases provide powerful tools for querying, sorting, and filtering data, making them ideal for data analysis and reporting.
- **Data security:** Relational databases offer a range of security features, such as encryption, access controls, and authentication, to protect sensitive data from unauthorized access.
- **Data integration:** Relational databases are capable of integrating with other systems and applications, allowing organizations to easily share and exchange data.

Chapter 4

Key-Value Based Databases

4.1 Introduction

Key-value based databases[7] are a type of database management system that uses a simple data model consisting of key-value pairs. In this model, data is stored in the form of key-value pairs, where the key is a unique identifier for a particular piece of data and the value is the data itself. Key-value based databases are known for their simplicity, scalability, and high performance.



Phone directory

Key	Value
Paul	(091) 9786453778
Greg	(091) 9686154559
Marco	(091) 9868564334

MAC table

Key	Value
10.94.214.172	3c:22:fb:86:c1:b1
10.94.214.173	00:0a:95:9d:68:16
10.94.214.174	3c:1b:fb:45:c4:b1

Figure 4.1: Key-Value based database (Redis)[8]

The history of key-value based databases can be traced back to the early days of computing, where simple data storage systems were used to store and retrieve data. In the 1960s, the term ”key-value store” was first used to describe a type of database management system that used a simple key-value data model.

Today, key-value based databases are widely used in a variety of applications, including web caching, content management systems, and distributed systems. Some popular key-value based databases include Memcached, Redis, and Amazon DynamoDB.

4.2 Types of Key-Value based Databases

There are several different types of key-value based databases, each with its own characteristics and features. The most common types of key-value based databases include:

- **In-memory key-value databases:** In-memory key-value databases are designed to store data in the computer's memory rather than on disk. This allows for fast access and updates to the data, making them ideal for applications that require high-speed data access. However, because the data is stored in memory, these databases are limited in the amount of data that they can store and are not suitable for storing large amounts of data.

- **Persistent key-value databases:** Persistent key-value databases are designed to store data on disk, allowing for larger amounts of data to be stored and accessed. These databases are ideal for applications that require data to be stored for long periods of time, such as in e-commerce applications. They are also suitable for applications that require data to be accessed and updated frequently, such as in social media applications.

- **Distributed key-value databases:** Distributed key-value databases are designed to store data across multiple servers, allowing for large amounts of data to be stored and accessed. These databases are ideal for applications that require high availability, scalability, and fault tolerance, such as in e-commerce and other web-based applications. They are also suitable for applications that require data to be accessed and updated frequently, such as in social media applications.

- **Column-oriented key-value databases:** Column-oriented key-value databases are designed to store data in columns rather than rows, allowing for efficient querying and data access. These databases are ideal for applications that require data to be accessed and updated frequently, such as in social media and other web-based applications. They are also suitable for applications that require data to be stored for long periods of time, such as in e-commerce applications.

4.3 Advantages and Disadvantages of Key-Value Databases

4.3.1 Advantages of Key-Value Databases

- **Simplicity:** Key-value databases[9] are easy to implement and maintain as they do not require a complex schema or data structure.

- **Scalability:** Key-value databases can easily handle large amounts of data and high levels of traffic without compromising performance.

- **High performance:** Key-value databases are designed to provide fast read and write speeds, making them suitable for applications that require real-time data access.

- **Flexibility:** Key-value databases allow for the storage of any data type, making them adaptable to changing data requirements.
- **Cost-effective:** Key-value databases are often more affordable than traditional relational databases, making them a cost-effective option for small and medium-sized businesses.

4.3.2 Disadvantages of Key-Value Databases

- **Limited querying capabilities:** Key-value databases do not support complex querying, making them less suitable for applications that require complex data analysis.
- **Limited data integrity:** Key-value databases do not enforce data integrity, which can lead to data inconsistencies and errors.
- **Lack of data relationships:** Key-value databases do not support the concept of data relationships, making it difficult to manage complex data structures.
- **Lack of transactions:** Key-value databases do not support transactions, making them less suitable for applications that require data consistency.

4.4 Applications of Key-Value Based Databases

- **Data Caching:** Key-value databases are often used to store cached data for quick retrieval. This can be beneficial for web applications that require fast access to frequently used data, such as user profiles or product information.
- **Content Management Systems (CMS):** Key-value databases are commonly used in CMS platforms to store and retrieve website content, such as articles and pages.
- **E-commerce:** Key-value databases are well-suited for e-commerce applications, as they can easily store and retrieve large amounts of data related to products, inventory, and customer information.
- **Real-time Analytics:** Key-value databases are often used to store and retrieve data for real-time analytics and reporting, as they provide fast access to data and can easily handle large amounts of data.
- **IoT (Internet of Things):** Key-value databases are often used in IoT applications to store and retrieve data from sensors and other devices, as they can handle large amounts of data and provide fast access to data.
- **Web Applications:** Key-value based databases are commonly used in web applications to store session information, user preferences, and other data that is accessed frequently. This allows web applications to quickly retrieve data and provide a smooth user experience.

- **Gaming:** Key-value based databases are also used in gaming applications to store player information, game state, and other data that is accessed frequently during gameplay. This allows games to quickly retrieve data and provide a seamless gaming experience.

Chapter 5

Document Based Databases

5.1 Introduction

A document-based database[10] is a type of database that is based on the storage and retrieval of documents. These documents are typically unstructured and do not conform to a pre-defined schema, allowing for a greater level of flexibility and scalability compared to traditional relational databases.

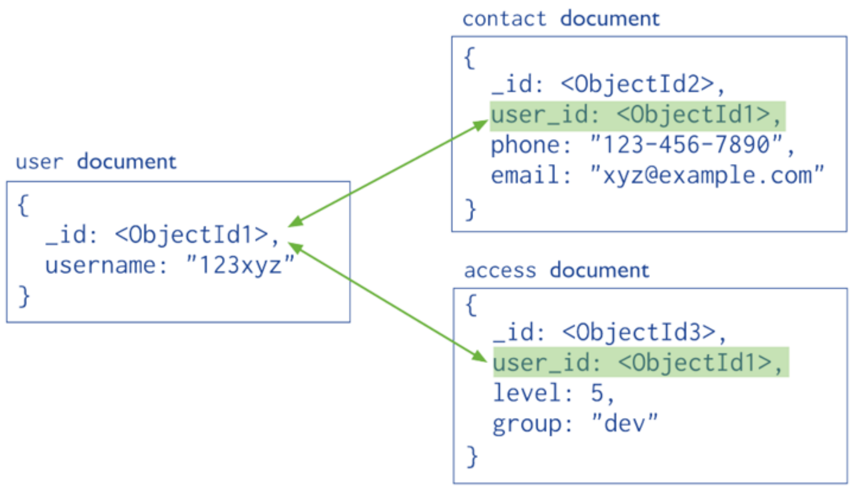


Figure 5.1: Document based database (MongoDB)[11]

One of the key features of document-based databases is their ability to store complex data structures, such as nested objects and arrays, without the need for pre-defined schemas. This allows for greater flexibility in data management, as new data can be easily added and stored without the need for complex schema updates.

In the early days of document-based databases, the most commonly used database system was the Document Management System (DMS), which was used primarily for the storage and retrieval of documents in the legal and financial industries. These systems were typically implemented on mainframe computers, and were designed to handle the large volumes of data generated by these industries.

In the 1990s, the emergence of the World Wide Web and the rise of the internet led to the development of new document-based database systems that were designed to handle the large volumes of data generated by web applications. These systems were typically implemented on distributed networks of servers, and were designed to handle the scalability and performance requirements of the web.

One of the most popular document-based database systems of this era was MongoDB, which was released in 2009. MongoDB was designed to be a scalable and high-performance database system that could handle the large volumes of data generated by web applications. It was built on the concept of a document-oriented database model, and was one of the first database systems to support the storage and retrieval of complex data structures without the need for pre-defined schemas.

Today, document-based databases are widely used in a variety of applications, including web applications, mobile applications, and big data analytics. They are commonly used in conjunction with other database systems, such as relational databases and graph databases, to provide a flexible and scalable data management platform.

5.2 Examples of Document Based Databases

- **MongoDB:** This is a popular NoSQL database that uses a document-oriented model. It allows for flexible schema design and easy scalability.
- **CouchDB:** This database uses a JSON-based document model and has a robust querying and indexing system. It also has built-in replication and synchronization capabilities.
- **Elasticsearch:** This is a search engine and analytics platform that uses a document-oriented model. It allows for real-time data indexing and search capabilities.
- **Firebase:** This is a cloud-based database that uses a JSON-based document model. It offers real-time data synchronization and automatic scaling capabilities.
- **Cosmos DB:** This is a globally distributed, multi-model database that supports document-oriented storage. It offers multiple consistency levels and automatic indexing of data.

5.3 Advantages and Disadvantages of Document Based Databases

5.3.1 Advantages of Document Based Databases

- **Flexibility:** Document-based databases allow for flexibility in the data structure, allowing users to store different types of data and information in a single document. This allows for easy customization and adaptation to changing business needs and requirements.
- **Scalability:** Document-based databases are scalable, allowing users to add more documents and data as their business grows and evolves.
- **High performance:** Document-based databases are designed to handle large volumes of data and high-speed data processing, making them ideal for businesses with high performance requirements.
- **Easy querying:** Document-based databases allow for easy querying of data using simple, intuitive query languages. This allows users to quickly access and retrieve the data they need, without the need for complex SQL queries.

5.3.2 Disadvantages of Document Based Databases

- **Limited data integrity:** Document-based databases do not have the same level of data integrity as traditional relational databases. This means that data may not be consistently formatted and may be prone to errors or inconsistencies.
- **Lack of data normalization:** Document-based databases do not follow the principles of data normalization, which can lead to data redundancy and wasted storage space.
- **Limited support for complex data relationships:** Document-based databases are not well-suited to complex data relationships, such as many-to-many relationships, which can be difficult to model and query.
- **Limited support for transactions:** Document-based databases do not support transactions, which can make it difficult to ensure data integrity and consistency in complex business processes.

5.4 Applications of Document based Databases

- **Content management systems:** Document-based databases are commonly used in content management systems (CMS) to store and manage large amounts of textual and multimedia content. Examples include WordPress, Drupal, and Joomla.

- **E-commerce websites:** Document-based databases are often used in e-commerce websites to store and manage customer information, product details, and order details. This allows for efficient retrieval and manipulation of data for various operations such as checkout and customer service.

- **Healthcare applications:** Document-based databases are used in healthcare applications to store and manage patient records, medical histories, and diagnostic results. This allows for efficient access and analysis of data for various purposes such as disease tracking and treatment planning.

- **Social media platforms:** Document-based databases are commonly used in social media platforms to store and manage user profiles, posts, and interactions. This allows for efficient retrieval and analysis of data for various purposes such as personalization and recommendation.

- **Data analytics and machine learning:** Document-based databases are often used in data analytics and machine learning applications to store and manage large amounts of structured and unstructured data. This allows for efficient access and manipulation of data for various purposes such as data exploration and model training.

Chapter 6

Graph Based Databases

6.1 Introduction

A graph database[12] is a type of database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. This is in contrast to the traditional relational database, which uses tables and rows to store data.

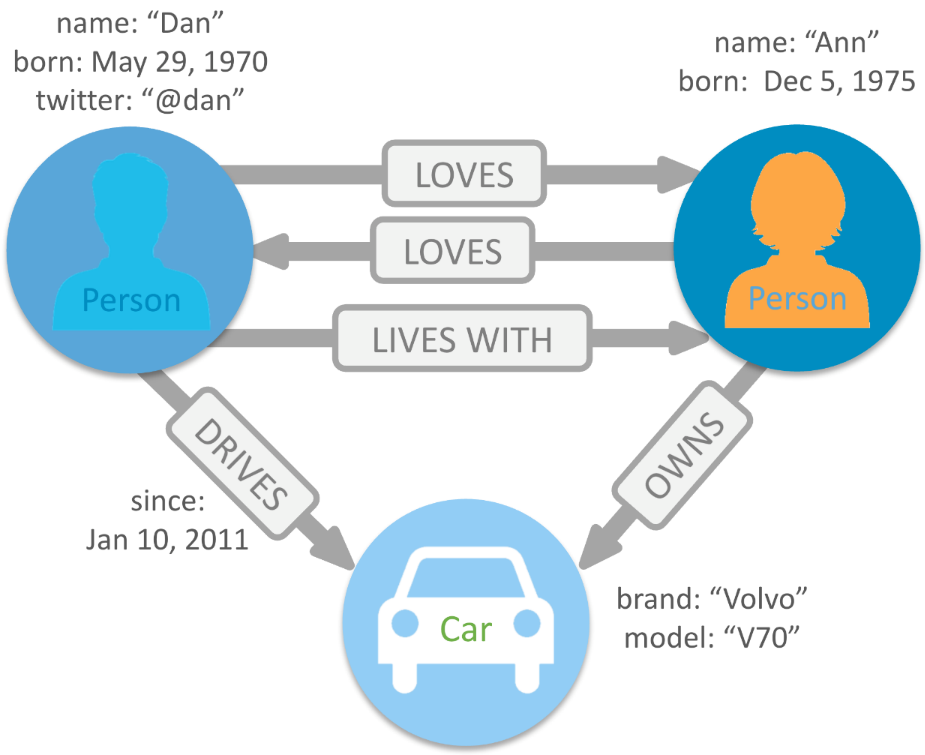


Figure 6.1: Graph based database (Neo4j) [13]

One of the key early applications of graph databases was in the field of artificial intelligence, where researchers used these databases to represent and reason about complex knowledge structures. In the 1980s and 1990s, a number of research groups and companies developed graph database systems for various applications, including expert systems, natural language processing, and geographic information systems.

The first commercial graph database, Neo4j, was released in 2007 and has since become one of the most popular graph databases in use today. In the years since the release of Neo4j, many other companies have developed their own graph databases, including TigerGraph, AWS Neptune, and Azure Cosmos DB.

The growth of the internet and the rise of big data have led to an increased interest in graph databases in recent years. These databases are well-suited for applications that involve complex relationships and interconnected data, such as social networks, recommendation engines, and fraud detection.

6.2 Types of Graph Based Databases

- **Property graph**: Property graph databases are the most common type of graph based database. They are a flexible and powerful data model that allows for the storage of nodes, edges, and properties in a graph structure. The nodes and edges in a property graph database represent the entities and relationships in the data, while the properties represent the attributes of those entities and relationships. This data model allows for complex and dynamic queries, as well as the ability to add new data easily. Some examples of property graph databases include Neo4j and JanusGraph.
- **Triple stores:** triple stores use a triple data model to represent data in a graph structure. In a triple store, the data is represented as subject-predicate-object triples, where the subject and object are the nodes in the graph, and the predicate is the relationship between those nodes. This data model is more rigid and structured than the property graph model, and is often used for data that is highly structured and has well-defined relationships. Some examples of triple stores include Apache Jena and Stardog.

6.3 Advantages and Disadvantages of Graph Based Databases

6.3.1 Advantages of Graph Based Databases

- **Flexible and Scalable:** Graph databases can handle large amounts of data and support multiple data types and relationships, allowing for complex queries and data exploration.

- **Fast and Efficient:** Graph databases use index-free adjacency, which allows for faster traversal and querying of data compared to other database models.
- **Intuitive and Easy to Use:** Graph databases use visual representations of data, making it easier for users to understand and work with the data.
- **Strong Data Integrity:** Graph databases support constraints and rules, ensuring the integrity and accuracy of the data.

6.3.2 Disadvantages of Graph Based Databases

- **Limited Support for Traditional SQL Queries:** Some graph databases do not support traditional SQL queries, making it difficult for users familiar with SQL to work with the data.
- **High Memory Requirements:** Graph databases often require large amounts of memory to store and process data, which can be expensive and challenging to manage.
- **Limited Support for Distributed Systems:** Some graph databases do not support distributed systems, making it difficult to scale and manage large amounts of data across multiple servers.
- **Limited Support for External Tools and Applications:** Some graph databases do not have strong integration with external tools and applications, limiting their usability and flexibility.

6.4 Applications of Graph based Databases

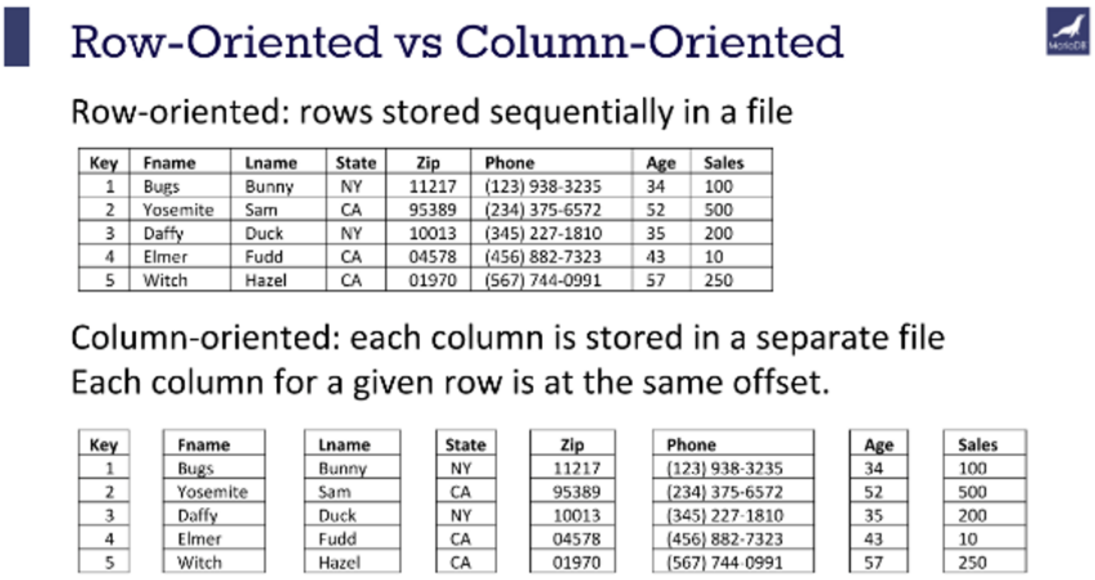
- **Social media networks:** Graph databases are often used to store and analyze connections between users on social media platforms, such as friend connections on Facebook or follower relationships on Twitter.
- **Fraud detection:** Graph databases can help identify patterns of fraudulent activity by analyzing connections between different entities, such as accounts, transactions, and locations.
- **Supply chain management:** Graph databases can be used to map out complex supply chain networks and optimize the flow of goods and materials.
- **Recommendation engines:** Graph databases can be used to generate personalized product or content recommendations based on user connections and preferences.
- **Knowledge management:** Graph databases can be used to store and organize large amounts of information and relationships between different pieces of data, making it easier to search and access knowledge.

Chapter 7

Column Based Databases

7.1 Introduction

A column-based database[14] is a type of database management system that stores data in columns rather than rows. This type of database is designed to provide fast access to large amounts of data by storing the data in a way that allows the database to quickly retrieve specific columns of data without having to read through the entire database.



Row-Oriented vs Column-Oriented

Row-oriented: rows stored sequentially in a file

Key	Fname	Lname	State	Zip	Phone	Age	Sales
1	Bugs	Bunny	NY	11217	(123) 938-3235	34	100
2	Yosemite	Sam	CA	95389	(234) 375-6572	52	500
3	Daffy	Duck	NY	10013	(345) 227-1810	35	200
4	Elmer	Fudd	CA	04578	(456) 882-7323	43	10
5	Witch	Hazel	CA	01970	(567) 744-0991	57	250

Column-oriented: each column is stored in a separate file
Each column for a given row is at the same offset.

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
CA
CA

Zip
11217
95389
10013
04578
01970

Phone
(123) 938-3235
(234) 375-6572
(345) 227-1810
(456) 882-7323
(567) 744-0991

Age
34
52
35
43
57

Sales
100
500
200
10
250

Figure 7.1: Column based database (MariaDB) [15]

Earlier, column-based databases became increasingly popular among businesses and organizations that needed to store and manage large amounts of data. Many of these organizations were attracted to the column-based model because it allowed them to quickly retrieve specific columns of data without having to read through the entire database.

In the late 1980s and early 1990s, the rise of the internet and the growth of e-commerce led to an explosion in the amount of data being generated and stored by businesses and organizations. This led to an increased demand for efficient and scalable database management systems, and column-based databases became even more popular as a result.

Today, column-based databases are used by a wide variety of organizations, including large enterprises, online retailers, financial institutions, and government agencies. These organizations rely on column-based databases to store and manage vast amounts of data, and to provide fast and efficient access to that data.

7.2 Examples of Column Based Databases

7.2.1 Hadoop Columnar Storage

- Uses HDFS to store large amounts of data
- Designed for data warehousing and big data analysis
- Allows for fast query processing and efficient data compression
- Uses HDFS to store large amounts of data
- Designed for data warehousing and big data analysis
- Allows for fast query processing and efficient data compression

7.2.2 Apache Cassandra

- Distributed and scalable database
- Supports column-oriented data storage and query processing
- Provides high availability and fault tolerance

7.2.3 Apache Parquet

- Column-oriented data storage format
- Designed for efficient data storage and query processing in distributed systems
- Used in big data analysis and data warehousing applications

7.2.4 Apache Accumulo

- Column-oriented database built on top of Apache Hadoop
- Provides fine-grained security and cell-level access control
- Used for storing and querying large amounts of structured and unstructured data.

7.3 Advantages and Disadvantages of Column Based Databases

7.3.1 Advantages of Column Based Databases

- **Improved query performance:** Column based databases[16] store data in columns rather than rows, which makes it easier and faster to search for specific data within a large dataset.
- **Reduced storage space:** Column based databases store data in a more compact format, which leads to reduced storage space requirements.
- **Enhanced data compression:** Column based databases use advanced data compression techniques to further reduce the storage space required for data.
- **Improved scalability:** Column based databases can easily scale up or down to accommodate changing data volumes and workloads.

7.3.2 Disadvantages of Column Based Databases

- **Limited support for complex data types:** Column based databases may not support complex data types such as nested objects or arrays.
- **Lack of support for transactional operations:** Column based databases may not support transactional operations such as ACID (Atomicity, Consistency, Isolation, Durability) compliance.
- **Limited compatibility with existing tools and applications:** Column based databases may not be compatible with existing tools and applications that are designed for traditional row-based databases.
- **High learning curve:** Column based databases may require specialized training and expertise to set up and maintain, which can be a disadvantage for organizations with limited technical resources.

7.4 Applications of Column based Databases

- **Data warehousing and business intelligence:** Column-based databases are commonly used for data warehousing and business intelligence applications as they provide faster querying and analysis of large amounts of data.
- **Online transaction processing (OLTP):** Column-based databases are often used in OLTP systems to improve the performance of complex queries and transactions.
- **Big data analytics:** Column-based databases are well-suited for handling large amounts of data, making them ideal for big data analytics applications.

- **Real-time data processing:** Column-based databases can provide real-time data processing capabilities, allowing for immediate analysis and response to events and changes in data.
- **Financial analysis:** Column-based databases are commonly used in the financial industry for analyzing financial data and making informed investment decisions.
- **Retail analysis:** Retail companies often use column-based databases to analyze customer behavior and sales data, enabling them to make data-driven decisions.
- **Marketing analysis:** Column-based databases are used in marketing applications to analyze customer data and identify trends and patterns that can be used to improve marketing strategies.

Chapter 8

Native XML Based Databases

8.1 Introduction

Native XML databases[17] are databases specifically designed to store and manage XML data. These databases have a long history, with the first native XML database being developed in the late 1990s.

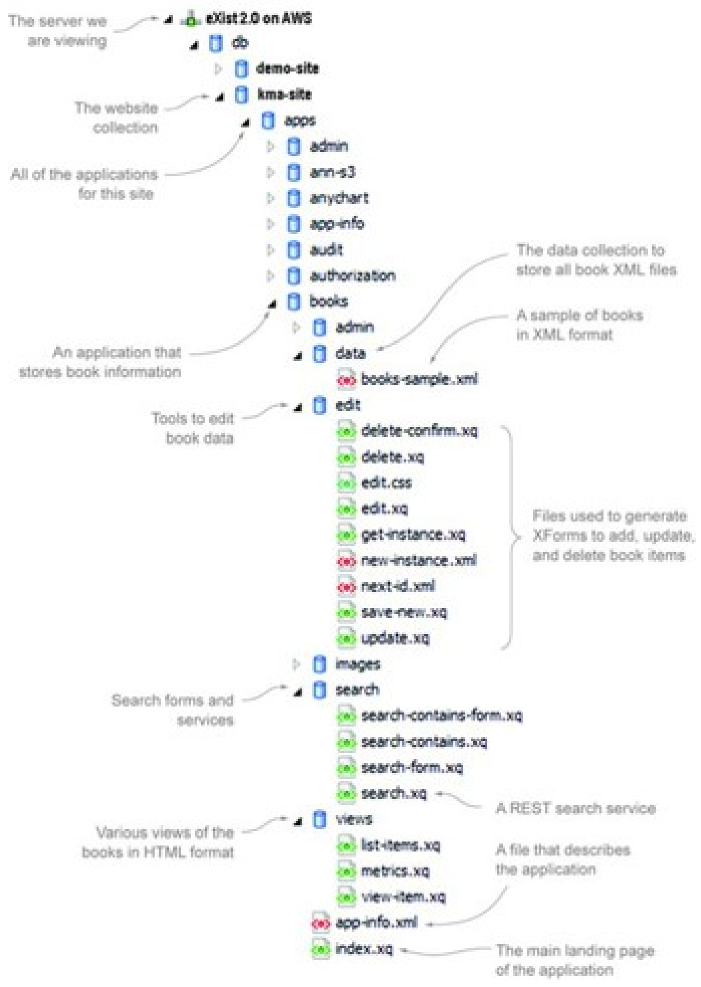


Figure 8.1: Native XML based database [18]

The first native XML database was called Tamino, and it was developed by Software AG. Tamino used a unique indexing and querying system that was optimized for handling XML data. This allowed for faster and more efficient processing of XML data compared to traditional relational databases.

In the early 2000s, several other native XML databases were developed, including eXist and BaseX. These databases further improved upon the capabilities of Tamino, adding support for more complex queries and advanced indexing systems.

Native XML databases became increasingly popular in the mid-2000s as the use of XML data grew in various industries. These databases were used in a variety of applications, including web services, content management systems, and data integration.

In recent years, native XML databases have continued to evolve, adding support for new technologies such as cloud computing and big data. Many of these databases now offer advanced features such as data security, data governance, and scalability.

Overall, the history of native XML databases has been one of constant innovation and development. These databases have played a crucial role in the rise of XML data, and they continue to be a valuable tool for managing and storing XML data.

8.2 Examples of Native XML Based Databases

- eXist
- BaseX
- Sedna
- Berkeley DB XML
- MarkLogic
- X-Hive/DB
- Qizx/Open
- Stylus Studio
- Apache Cocoon
- QFS XML DB

8.3 Advantages and Disadvantages of Native XML Based Databases

8.3.1 Advantages of Native XML Based Databases

- **Flexibility:** Native XML databases allow for the storage and manipulation of unstructured and semi-structured data in a flexible manner.

- **Standards-based:** XML is a widely-used, standardized format for representing data, making it easier to integrate with other systems and applications.
- **Query capabilities:** Native XML databases provide powerful query languages such as XQuery and XPath for searching and extracting data from the database.

8.3.2 Disadvantages of Native XML Based Databases

- **Performance:** Native XML databases can have slower performance compared to traditional relational databases, especially for large amounts of data.
- **Complexity:** XML is a complex language, and working with native XML databases can be challenging for users without a strong technical background.
- **Limited support:** Not all database management systems support native XML databases, so compatibility can be an issue.

8.4 Applications of Native XML based Databases

- Storing and managing large volumes of structured data, such as customer information, financial transactions, or healthcare records
- Enabling real-time data exchange and integration with other systems and applications through XML-based APIs
- Providing efficient querying and indexing capabilities for complex data structures and relationships
- Supporting standards-based data interoperability and semantic interoperability across different organizations and domains
- Facilitating flexible and dynamic schema evolution and schema-less data modeling
- Enabling robust data security and access control through XML-based authentication and authorization mechanisms.

Chapter 9
Hierarchical Databases

9.1 Introduction

A hierarchical database[19] is a type of database model that stores data in a tree-like structure. In this model, data is organized into a series of parent-child relationships, where each parent record can have multiple child records. Hierarchical databases were first developed in the 1960s and were widely used in mainframe computer systems. Today, hierarchical databases are used in a variety of applications, such as telecommunications, inventory management, and airline reservations.

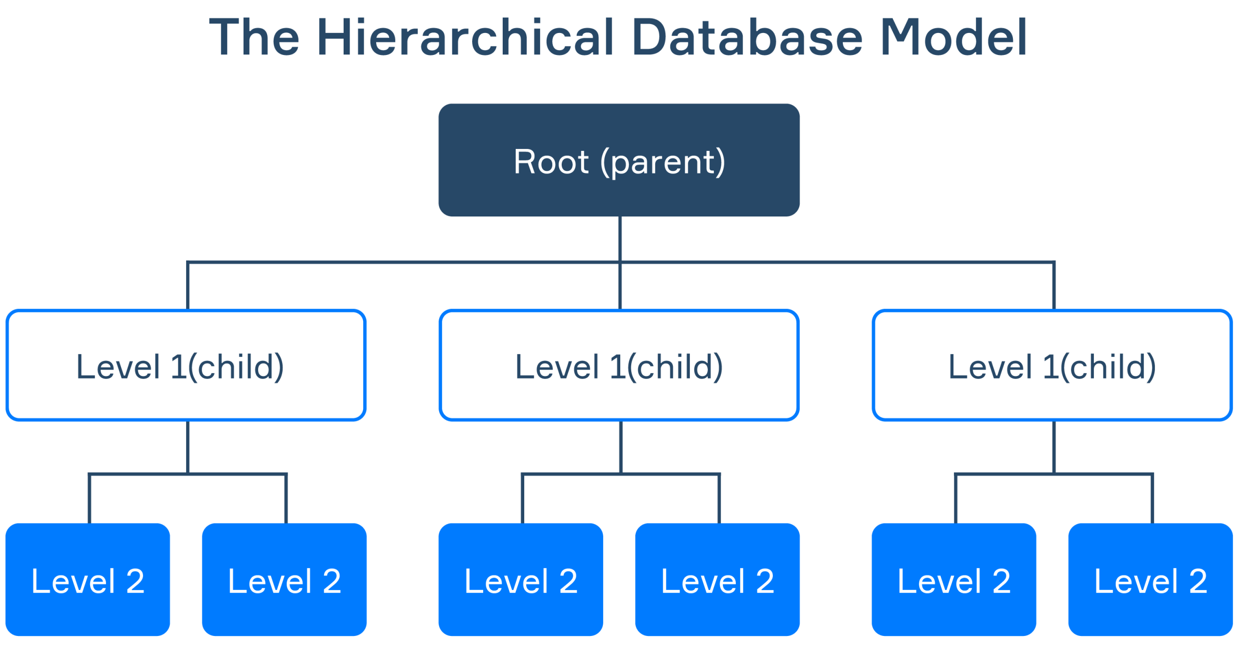


Figure 9.1: Hierarchical Databases [19]

Hierarchical databases were first introduced in the early 1960s as a way to store and manage data on early computer systems. IBM was the first to introduce this model with their Information Management System (IMS), which became widely used in the banking and airline industries. However, as computing technology evolved, hierarchical databases became less popular, and new models such as the relational database model were introduced.

Hierarchical databases were designed to meet the needs of early computer systems, which had limited processing power and storage capabilities. These systems needed a way to efficiently store and manage large amounts of data, and hierarchical databases provided a simple, hierarchical structure that was easy to manage.

9.2 Examples of Hierarchical Databases

- Filesystems
- DNS
- LDAP directories

9.3 Advantages and Disadvantages of Hierarchical Databases

9.3.1 Advantages of Hierarchical Databases

- **Simple**: Hierarchical databases have a simple structure that is easy to understand and manage.
- **Fast**: Hierarchical databases are fast and efficient at retrieving data, as they use a tree-like structure that allows for quick navigation.
- **Low overhead**: Hierarchical databases have low overhead, as they do not require complex queries or indexing.
- **Efficient storage**: Hierarchical databases are efficient in terms of storage, as they store related data together in one record.

9.3.2 Disadvantages of Hierarchical Databases

- **Limited flexibility:** Hierarchical databases are limited in terms of flexibility, as they are designed to store data in a specific structure.
- **Limited scalability:** Hierarchical databases can become complex and difficult to manage as the number of records and relationships increases.
- **Limited query capability:** Hierarchical databases have limited query capability, as they only allow for simple navigational queries.

9.4 Applications of Hierarchical Databases

- **Large mainframe systems:** Hierarchical databases are still used in some large mainframe systems, such as those used in the banking and airline industries.
- **Manufacturing:** Hierarchical databases are used in manufacturing to manage bills of materials and production schedules.
- **Inventory management:** Hierarchical databases are used in inventory management to track the movement of goods and manage stock levels.
- **Library systems:** Hierarchical databases are used in library systems to manage books, patrons, and loans.

Chapter 10
Network databases

10.1 Introduction

A network database[20] is a type of database model that stores data in a graph-like structure, where each record can have multiple parent and child records. This model was introduced in the late 1960s as an improvement over the hierarchical model, and it became widely used in the 1970s and 1980s. Today, network databases are less popular than other models, but they are still used in some applications.

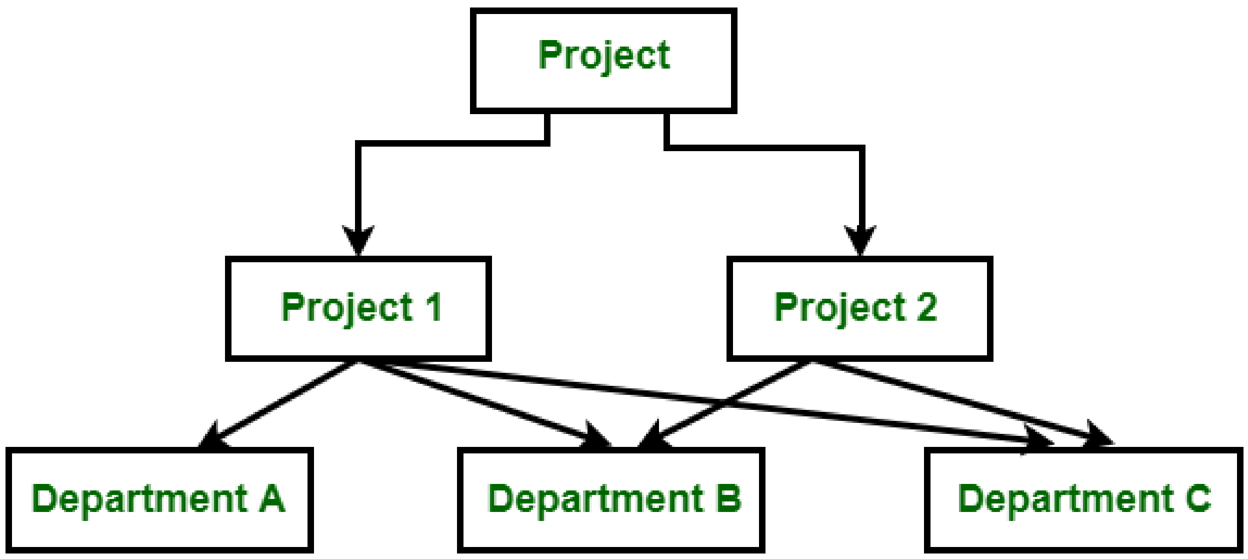


Figure 10.1: Network database [20]

The network database model was first introduced in the late 1960s as an improvement over the hierarchical model. The CODASYL (Conference on Data Systems Languages) Data Base Task Group developed the first specification for the network database model in 1969. The model was widely used in the 1970s and 1980s, particularly in applications such as banking and airline reservation systems. However, as the relational model gained popularity in the 1990s, the use of network databases declined.

The network database model was developed to address some of the limitations of the hierarchical model. In the hierarchical model, records are organized in a tree-like structure with one parent and multiple children, which limits the flexibility of the data structure. The network model allows records to have multiple parents and children, providing greater flexibility in representing relationships between data.

10.2 Advantages and Disadvantages of Network databases

10.2.1 Advantages of Network databases

- **Flexibility:** The network model allows records to have multiple parent and child records, providing greater flexibility in representing relationships between data.
- **Efficient retrieval:** The network model allows for efficient retrieval of data, as records can be accessed directly through relationships with other records.
- **Data integrity:** The network model allows for complex data relationships to be enforced through referential integrity constraints.

10.2.2 Disadvantages of Network databases

- **Complexity:** The network model can be complex to design and manage, particularly as the number of records and relationships increases.
- **Limited scalability:** Network databases can become difficult to manage as the number of records and relationships increases.
- **Limited adoption:** Network databases are less popular than other models, which can limit the availability of tools and support.

10.3 Applications of Network databases

- **Banking:** Network databases are used in banking systems to manage customer accounts and transactions.
- **Airline reservations:** Network databases are used in airline reservation systems to manage flight schedules, reservations, and ticketing.
- **Manufacturing:** Network databases are used in manufacturing systems to manage bills of materials and production schedules.
- **Scientific research:** Network databases are used in scientific research to manage data relationships in fields such as biology and genetics.

Chapter 11

Time series databases

11.1 Introduction

A time series database[21] is a type of database designed to handle large amounts of data with timestamps, typically generated from sensors, IoT devices, or other time-stamped events. Time series databases are optimized for querying and analyzing data in chronological order, making them ideal for applications that require real-time data analysis and forecasting.

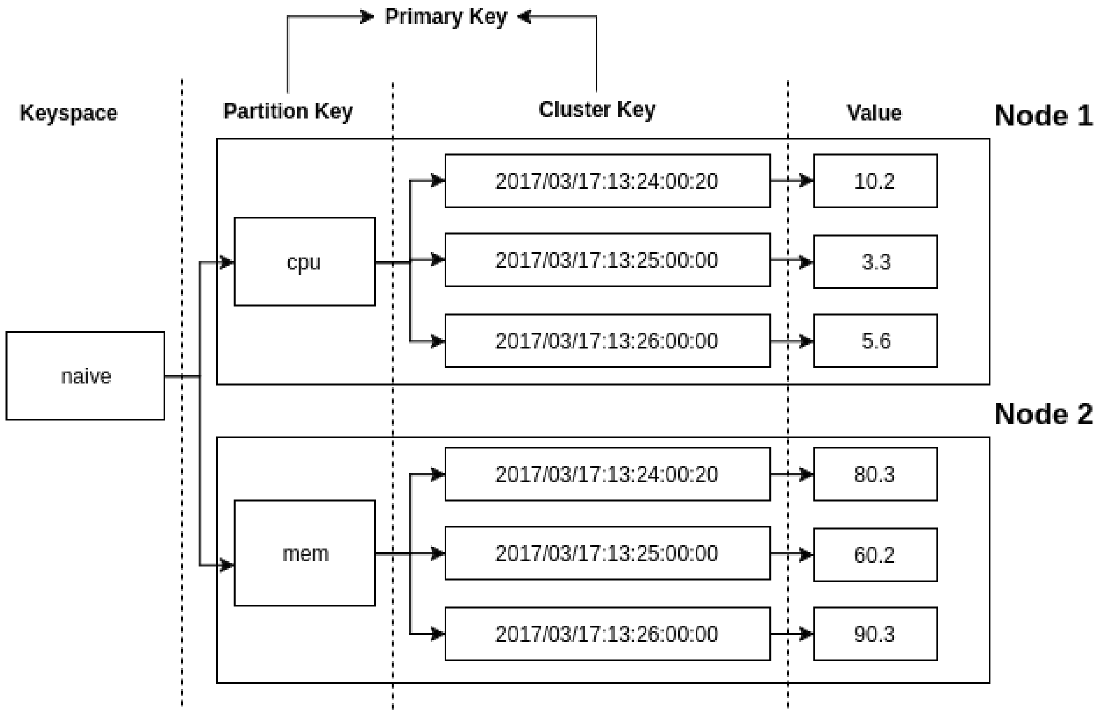


Figure 11.1: Time series database [21]

Time series databases have been used for decades, with early implementations found in manufacturing and engineering applications. However, with the rise of IoT devices and the increasing importance of real-time data analysis in industries such as finance, healthcare, and logistics, the demand for time series databases has grown significantly in recent years.

The need for time series databases stems from the challenges associated with storing and analyzing large amounts of time-stamped data. Traditional databases are not optimized for querying and analyzing data in chronological order, which can result in slow query times and high storage costs. Time series databases address these challenges by providing a specialized data model and indexing scheme that allows for fast and efficient querying of time-series data.

11.2 Examples of Time series databases

- OpenTSDB
- Prometheus
- InfluxDB
- TimescaleDB

11.3 Advantages and Disadvantages of Time series databases

11.3.1 Advantages of Time series databases

- **Fast querying:** Time series databases are optimized for querying and analyzing time-stamped data, which results in faster query times compared to traditional databases.
- **Efficient storage:** Time series databases use specialized data models and indexing schemes that are designed to optimize storage of time-stamped data.
- **Real-time data analysis:** Time series databases enable real-time data analysis and forecasting, which is critical in applications such as finance, healthcare, and logistics.

11.3.2 Disadvantages of Time series databases

- **Limited flexibility:** Time series databases are designed specifically for handling time-stamped data and may not be well-suited for applications that require more complex data structures.
- **Limited tool support:** While there are several time series databases available, they may not have the same level of tool support and integration as traditional databases.

- **High learning curve:** Time series databases have a specialized data model and indexing scheme, which may require a significant learning curve for users unfamiliar with the technology.

11.4 Applications of Time series databases

- **IoT:** Time series databases are used to store and analyze data generated by IoT devices, such as temperature sensors, smart meters, and security cameras.
- **Finance:** Time series databases are used in finance applications to analyze stock prices, currency exchange rates, and other financial data.
- **Healthcare:** Time series databases are used in healthcare applications to analyze patient data, such as vital signs and medication usage.
- **Logistics:** Time series databases are used in logistics applications to track and analyze shipment data, such as delivery times and inventory levels.

Chapter 12

NewSQL databases

12.1 Introduction

NewSQL[22] is a relational database management system that seeks to combine the scalability and flexibility of NoSQL systems with the consistency and reliability of traditional SQL databases. It emerged in response to the limitations of traditional SQL databases in handling large-scale data processing, particularly in web applications that require high transaction rates.

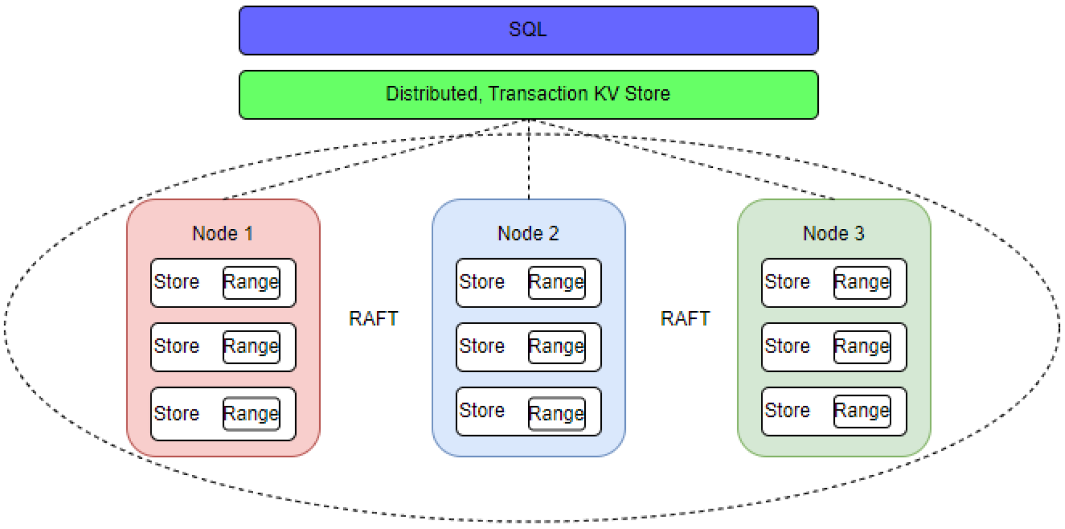


Figure 12.1: NewSQL database [22]

NewSQL databases were first introduced in 2009 by several researchers who aimed to build a system that could handle both online transaction processing (OLTP) and online analytical processing (OLAP) workloads in a single platform. Since then, several NewSQL databases have been developed, including NuoDB, TiDB, and CockroachDB.

Traditional SQL databases struggle to keep up with the needs of modern applications that require scalability, availability, and high-performance processing of large datasets.

NewSQL databases address these needs by providing the best of both worlds: the consistency and reliability of traditional SQL databases, and the scalability and flexibility of NoSQL systems.

12.2 Examples of NewSQL databases

- MemSQL
- VoltDB
- Spanner
- Calvin
- CockroachDB
- FaunaDB
- yugabyteDB
- PlanetScale

12.3 Advantages and Disadvantages of NewSQL databases

12.3.1 Advantages of NewSQL databases

- **Scalability:** NewSQL databases can scale horizontally, allowing for distributed architectures that can handle high volumes of data and transactions.
- **Performance:** NewSQL databases use in-memory processing and distributed computing to provide high-performance processing of large datasets.
- **Consistency:** NewSQL databases provide strong consistency guarantees, ensuring that data is always consistent and accurate, even in distributed environments.
- **Flexibility:** NewSQL databases can handle both OLTP and OLAP workloads, making them suitable for a wide range of applications.

12.3.2 Disadvantages of NewSQL databases

- **Complexity:** NewSQL databases can be complex to set up and manage, requiring specialized skills and knowledge.
- **Cost:** NewSQL databases can be more expensive than traditional SQL databases, particularly when scaling out to handle large datasets.

12.4 Applications of NewSQL databases

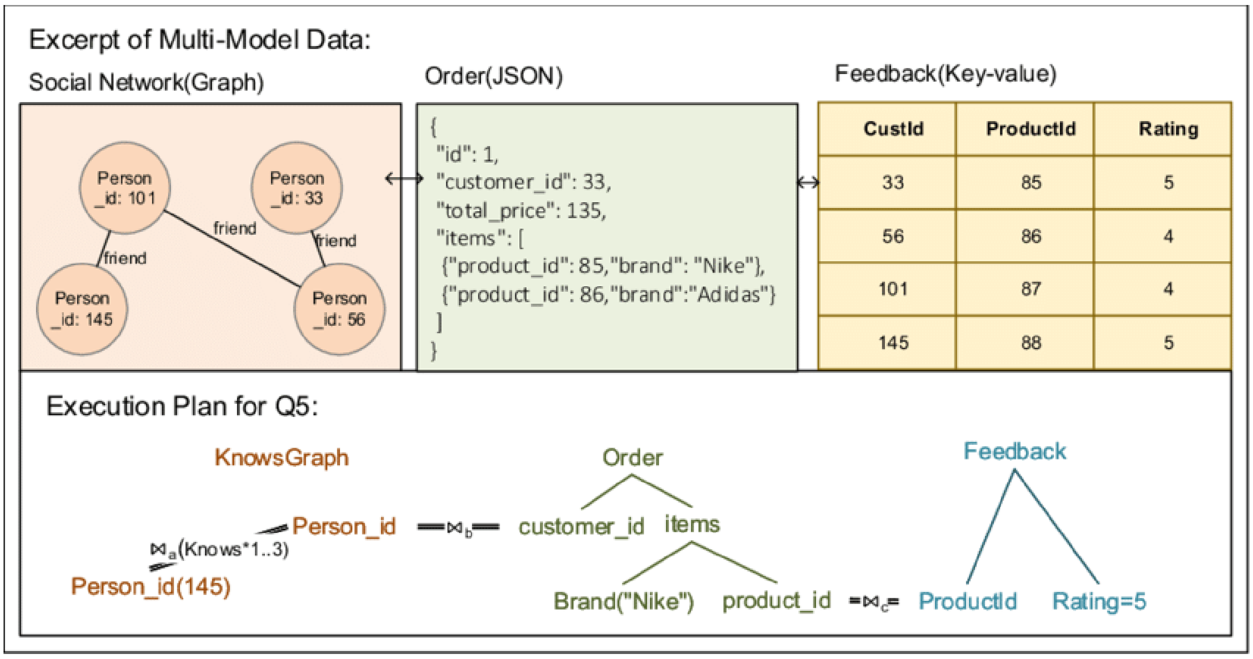
- **E-commerce platforms:** NewSQL databases can handle high transaction rates and large volumes of data, making them well-suited for e-commerce applications.
- **Financial services:** NewSQL databases can provide fast and reliable processing of financial data, making them ideal for financial services applications.
- **Healthcare:** NewSQL databases can handle large volumes of healthcare data, such as patient records and medical imaging data, making them well-suited for healthcare applications.
- **Online gaming:** NewSQL databases can provide fast and reliable processing of game data, making them ideal for online gaming applications.

Chapter 13

Multi-model databases

13.1 Introduction

Multi-model databases[23] are a type of database management system that supports multiple data models within a single database. They provide users with the flexibility to store and manage different types of data using different models, such as relational, document-oriented, graph, or key-value. This allows for greater efficiency and versatility in data management.



```
{
 "id": 1,
 "customer_id": 33,
 "total_price": 135,
 "items": [
  {"product_id": 85,"brand": "Nike"},
  {"product_id": 86,"brand":"Adidas"}
 ]
}
```

CustId	ProductId	Rating
33	85	5
56	86	4
101	87	4
145	88	5

Figure 13.1: Multi-model database [23]

Multi-model databases have emerged in response to the need for more flexible and adaptable databases that can handle the diverse and complex data types and structures of modern applications. The concept of multi-model databases has been around for some time, but it wasn't until the advent of NoSQL databases that the idea gained more traction and popularity.

Traditional databases were designed to handle structured data using the relational model. However, with the rise of big data and the Internet of Things (IoT), databases need to handle unstructured and semi-structured data as well. Multi-model databases were created to address this need by allowing users to store and manage diverse data types using different models, while also providing a unified view of the data.

13.2 Examples of Multi-model databases

- ArangoDB
- OrientDB
- Couchbase

13.3 Advantages and Disadvantages of Multi-model databases

13.3.1 Advantages of Multi-model databases

- **Flexibility:** Multi-model databases allow users to store and manage different data types using different models, which provides greater flexibility and adaptability in data management.
- **Scalability:** Multi-model databases are designed to handle large volumes of data, making them highly scalable.
- **Performance:** Multi-model databases are optimized for specific data models, which can lead to improved performance for specific types of queries.
- **Integration:** Multi-model databases can integrate with different data sources and systems, which makes it easier to consolidate data from multiple sources.

13.3.2 Disadvantages of Multi-model databases

- **Complexity:** Managing multiple data models within a single database can be complex and require specialized knowledge and expertise.
- **Cost:** Multi-model databases can be more expensive than single-model databases, especially when it comes to licensing and maintenance.
- **Security:** Multi-model databases may be more vulnerable to security breaches, as multiple models and data types increase the complexity of the system.

13.4 Applications of Multi-model databases

- **Financial services:** Multi-model databases are used to manage large volumes of financial data, including transaction data, customer data, and market data.
- **Healthcare:** Multi-model databases are used to manage medical records, patient data, and clinical trial data.
- **E-commerce:** Multi-model databases are used to manage product catalogs, customer data, and order data.
- **IoT:** Multi-model databases are used to manage the vast amounts of data generated by IoT devices, including sensor data, machine data, and event data.

Chapter 14

Compatitive analysis of different database storage models

Table 14.1: Comparison between different database storage models

Database	Speed	Storage	Use Case	Community Support	OLTP / OLAP Support	Query Language	Examples
Flat File	Slowest (Read)	Unlimited	File Systems	High	N/A	N/A	CSV, XML
Relational	Slow (Write)	Small	Transactional Databases	High	OLTP and OLAP	SQL	MySQL
Key-Value	Fastest (Read)	Smallest	Authentication	Medium	OLAP	No Specific Query Language	JSON, Redis
Document	Fast (Read)	Relatively high	Single Entity queries	High	OLTP and OLAP	Document-oriented query languages (e.g. MongoDB query language)	MongoDB
Graph	Fastest (relationship)	Small	Social Network	Low	OLAP	Graph query languages (e.g. Cypher, Gremlin)	Neo4j, OrientDB
Column	Fast (Write)	Highest	Real-time analytics	Medium	OLAP	SQL,HQL,PIG	Cassandra, HBase
Native XML	Fast (XML)	High	Hierarchical data, websites	Low	N/A	XML Query languages (e.g. XPath, XQuery)	eXist, BaseX, X-Hive
Hierarchical	Fast (Tree)	Small	Legacy Applications	Low	N/A	DL/1	DNS, LDAP, IMS
Network	Fast (Tree+Network)	Small	Complex Relationships, Graphs, and Trees	Low	N/A	CODASYL (SQL Like)	Integrated Database Management System (IDMS)
Time series	Very Fast (logs)	High	Time-Stamped Data, Sensor data	Medium	OLAP	Based on Database (SQL like)	InfluxDB, TimescaleDB
NewSQL	Fast (Distributed)	High	High-Volume, High-Velocity Transactions	Medium	OLTP/OLAP	SQL	Google Cloud Spanner, MemSQL
Multi-model	Fast (Different Databases)	High	Multiple Data Models, Complex Data	Medium	OLTP/OLAP	Varies by Data Model	ArangoDB, Couchbase, OrientDB

Chapter 15
Conclusion

The conclusion of our discussion on various database storage models is that there are multiple options available for organizing and storing data in a database. Each of these models has its own strengths and weaknesses, and the appropriate model to use depends on the specific requirements and characteristics of the data and the application it is being used for.

The flat file model is the most basic and simple form of database storage, where data is organized in a single file without any structural relationships between the data elements. This model is easy to implement and can be used for small and simple datasets, but it lacks the ability to handle complex data relationships and is not suitable for large datasets.

The relational model is a more advanced and powerful database storage model, where data is organized into tables and relationships are established between the tables using primary and foreign keys. This model allows for the efficient and flexible management of complex data relationships, but it requires a high level of expertise to design and implement.

The key-value based model is a database storage model that uses a simple and efficient key-value pair mechanism to store data. This model is suitable for fast and efficient access to data, but it is not suitable for complex data relationships and queries.

The document based model is a database storage model that stores data in the form of documents, such as XML or JSON documents. This model allows for the efficient storage and retrieval of complex data structures, but it is not suitable for complex queries and data relationships.

The graph based model is a database storage model that stores data in the form of nodes and edges, representing entities and relationships between entities. This model is suitable for representing and managing complex data relationships, but it requires specialized knowledge and tools to design and implement.

The column based model is a database storage model that stores data in columns rather than rows, allowing for efficient querying of large datasets. This model is suitable for handling large datasets and complex queries, but it requires specialized knowledge and tools to design and implement.

The native XML database is a database storage model that is specifically designed to store and manage XML data. This model allows for the efficient and flexible management of complex data structures, but it is not suitable for all types of data and applications.

Hierarchical databases have played an important role in the development of database technology. While they have been largely replaced by relational databases in modern systems, they continue to be used in certain industries and legacy applications. Understanding their advantages, disadvantages, and uses can help businesses make informed decisions about their data management needs.

Network databases have played an important role in the development of database technology. Although they have been largely replaced by relational databases in modern systems, they continue to be used in certain industries and legacy applications. Understanding their advantages, disadvantages, and uses can help businesses make informed decisions about their data management needs.

Time series databases are an important tool for handling time-based data in various applications. They offer high performance, scalability, and real-time analytics capabilities, making them ideal for IoT, big data, and financial trading applications. While there are some limitations to their use, understanding their advantages, disadvantages, and uses can help businesses make informed decisions about their data management needs.

NewSQL databases are a relatively new but growing category of database management systems that offer the scalability and performance of NoSQL databases with the ACID compliance of traditional relational databases. They are ideal for high-transaction applications that require real-time processing and analysis, and they can handle both structured and unstructured data. While they have some limitations, understanding their advantages, disadvantages, and uses can help businesses make informed decisions about their data management needs.

Multi-model databases offer flexibility and efficiency in handling multiple data models within a single database instance. They are useful in complex applications that require different types of data to be stored and queried in the same system. While they have some disadvantages, such as increased complexity and limited support for some data models or query languages, they offer many advantages, including scalability, availability, and improved support for complex data relationships and analytics. Understanding the advantages, disadvantages, and uses of multi-model databases can help businesses make informed decisions about their data management needs.

Finally, the various database storage models provide different options for organizing and storing data in a database, and the appropriate model to use depends on the specific requirements and characteristics of the data and the application it is being used for.

Chapter 16
Future Work

In the future, a technical comparison of various database storage models will provide valuable insights into their relative strengths and weaknesses. This comparison will be based on various metrics, such as performance, scalability, data integrity, query capabilities, and compatibility with different programming languages and frameworks.

In the future, studying the current researches and new technologies being implemented on various database storage models will provide valuable insights into the latest trends and developments in this field. This will involve studying the research papers and technical reports published by leading experts and organizations, as well as attending conferences and workshops focused on database technologies.

Chapter 17

References

[1] S. Acharya, *Demystifying Nosql*. Wiley, 2019.

[2] D. Sullivan, *NoSQL for mere mortals*. Addison-Wesley, 2015.

[3] M. K. Yusof and M. Man, "Efficiency of flat file database approach in data storage and data extraction for big data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 9, no. 2, pp. 460–473, 2018.

[4] *Csv and excel*. [Online]. Available: `https://nodegoat.net/guide.s/22/upload-a-csv-file` (visited on 11/02/2022).

[5] G. A. Schreiner, D. Duarte, and R. d. S. Mello, "When relational-based applications go to nosql databases: A survey," *Information*, vol. 10, no. 7, p. 241, 2019.

[6] *What is a database model*. [Online]. Available: `https://www.lucidchart.com/pages/database-diagram/database-models` (visited on 11/03/2022).

[7] W. Puangsaijai and S. Puntheeranurak, "A comparative study of relational database and key-value database for big data applications," in *2017 International Electrical Engineering Congress (iEECON)*, IEEE, 2017, pp. 1–4.

[8] *What is a key-value database?* 2021. [Online]. Available: `https://redis.com/nosql/key-value-databases` (visited on 11/03/2022).

[9] R. Sánchez-de Madariaga, A. Muñoz, A. L. Castro, O. Moreno, and M. Pascual, "Executing complexity-increasing queries in relational (mysql) and nosql (mongodb and exist) size-growing iso/en 13606 standardized ehr databases," *JoVE (Journal of Visualized Experiments)*, no. 133, e57439, 2018.

[10] J. Han, E. Haihong, G. Le, and J. Du, "Survey on nosql database," in *2011 6th international conference on pervasive computing and applications*, IEEE, 2011, pp. 363–366.

[11] *Rules of data conversion from document to relational databases*, 2020. [Online]. Available: `https://kariera.future-processing.pl/blog/rules-of-data-conversion-from-document-to-relational-databases` (visited on 11/02/2022).

[12] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–39, 2008.

[13] *What is a graph database? - developer guides.* [Online]. Available: `https://neo4j.com/developer/graph-database` (visited on 10/28/2022).

[14] R. Mehra, N. Lodhi, and R. Babu, "Column based nosql database, scope and future," *International Journal of Research and Analytical Reviews*, vol. 2, no. 4, pp. 105–113, 2015.

[15] MariaDB, *What is a columnstore index and why it is important?* 2019. [Online]. Available: `https://mariadb.com/resources/blog/why-is-columnstore-important` (visited on 11/04/2022).

[16] D. J. Abadi, P. A. Boncz, and S. Harizopoulos, "Column-oriented database systems," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1664–1665, 2009.

[17] R. Bača, M. Krátkỳ, I. Holubová, *et al.*, "Structural xml query processing," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–41, 2017.

[18] *Chapter 5. native xml databases · making sense of nosql.* [Online]. Available: `https://livebook.manning.com/book/making-sense-of-nosql/chapter-5` (visited on 11/04/2022).

[19] *What is a hierarchical database?* 2021. [Online]. Available: `https://hyperskill.org/learn/step/17042` (visited on 03/05/2023).

[20] *Difference between network and relational data model*, 2021. [Online]. Available: `https://www.geeksforgeeks.org/difference-between-network-and-relational-data-model/` (visited on 03/05/2023).

[21] *Introduction to time series database*, 2021. [Online]. Available: `https://blog.dongyueweb.com/introduction_to_time_series_database.html` (visited on 03/06/2023).

[22] E. Pina, F. Sá, and J. Bernardino, "Newsql databases assessment: Cockroachdb, mariadb xpand, and voltdb," *Future Internet*, vol. 15, no. 1, p. 10, 2022. DOI: `10.3390/fi15010010`.

[23] C. Zhang and J. Lu, "Holistic evaluation in multi-model databases benchmarking," *Distributed and Parallel Databases*, vol. 39, no. 1, 1–33, 2019. DOI: `10.1007/s10619-019-07279-6`.