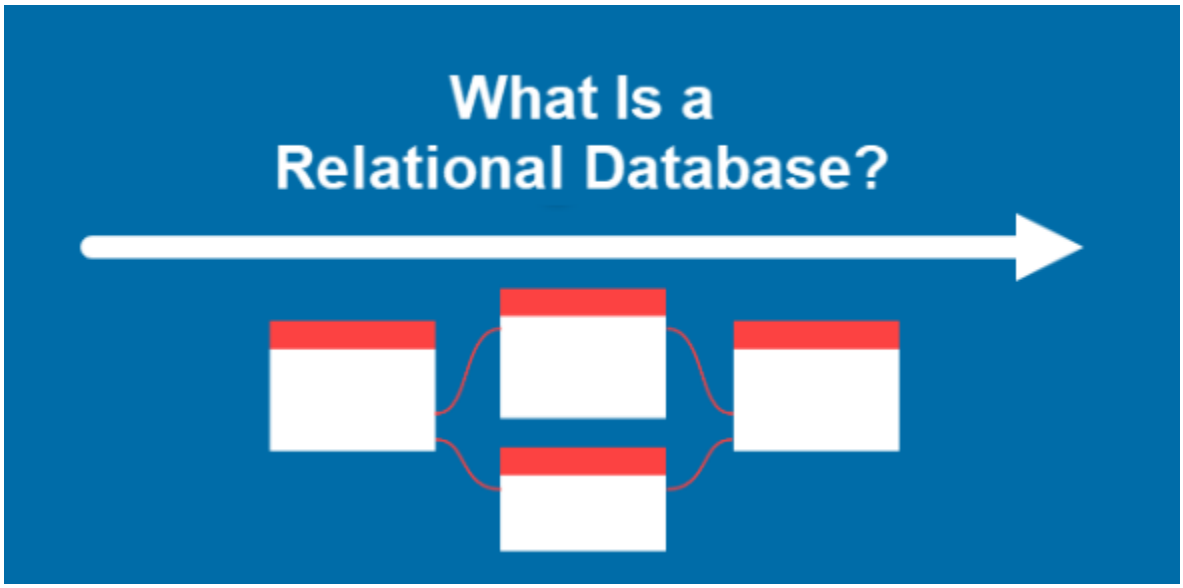
Submitted by: JAIR ADRAIAN T. VILLARANTE
3 - BSCS - A

CS 312
Applications Development and Emerging Technologies
Professor: Joseph C. Lorilla

TYPES OF DATABASE

Relational Database

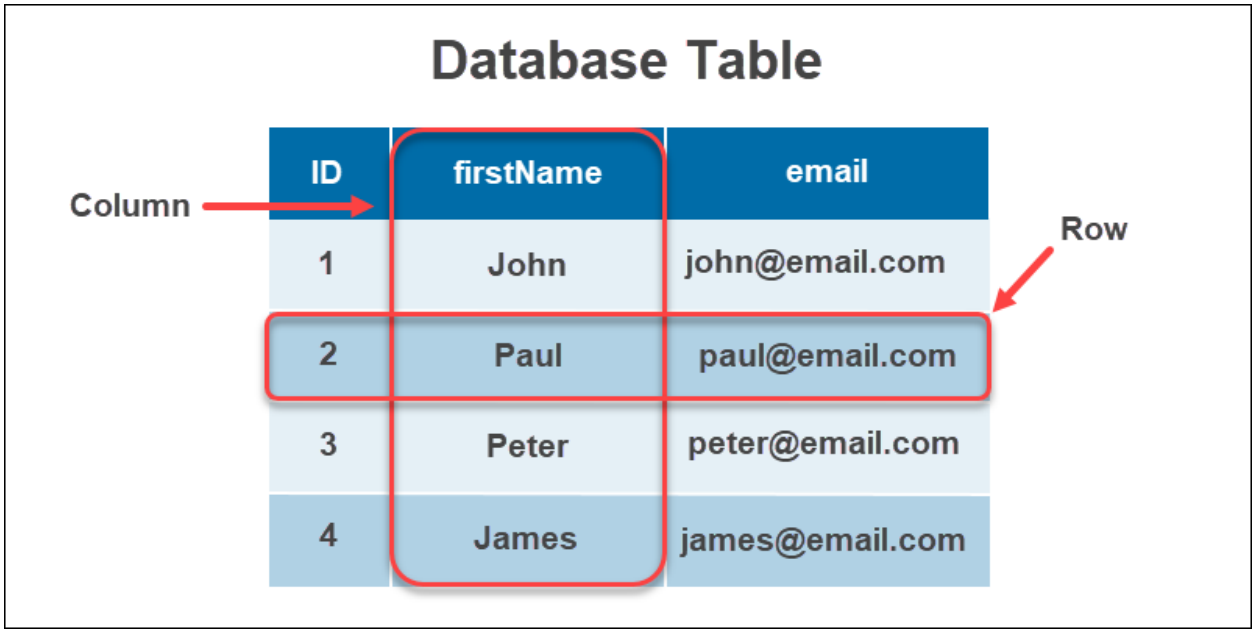


- A **relational database** is a type of database that focuses on the relation between stored data elements. It allows users to establish links between different sets of data within the database and use these links to manage and reference related data.
- Many relational databases use **SQL** (Structured Query Language) to perform queries and maintain data.

The **relational database** model is the most extensively used as well as the oldest database type. The three critical components of a relational database are:

- **Tables.** An entity type with relations.
- **Rows.** Records or instances of an entity type.

- **Columns.** Value attributes of instances.

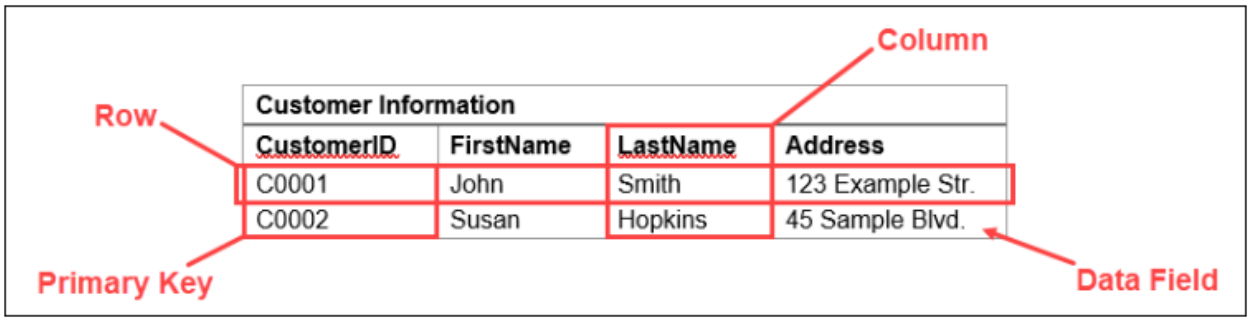


A relational database provides a set of data rows in response to a **query**. A query language, most commonly the Structured Query Language or **SQL**, helps create these data views.

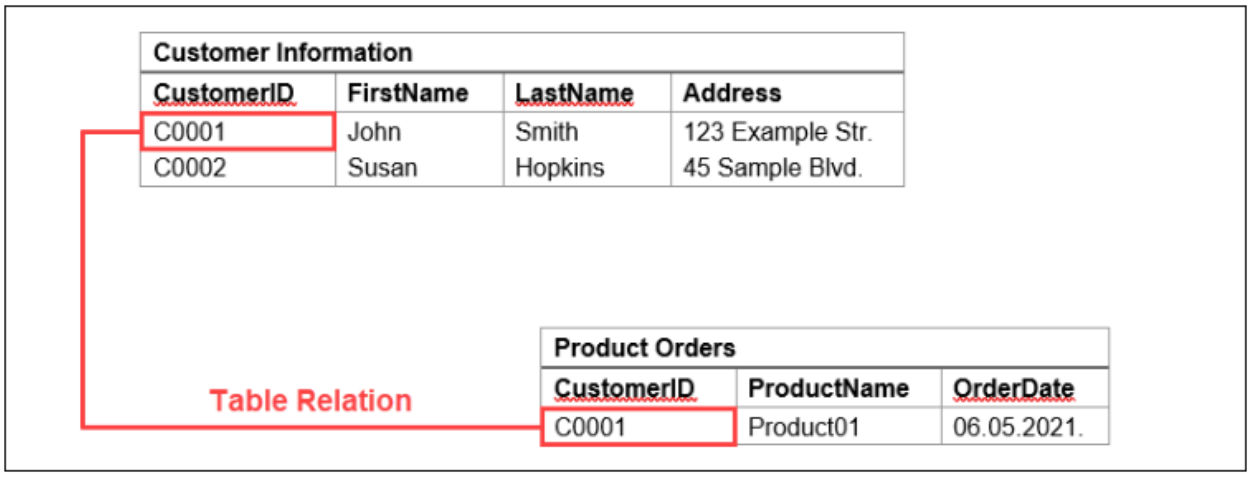
How Is Data in a Relational Database System Organized?

Relational database systems use a model that organizes data into tables of rows (also called *records* or *tuples*) and columns (also called *attributes* or *fields*). Generally, columns represent categories of data, while rows represent individual instances.

Let's use a digital storefront as an example. Our database might have a table containing customer information, with columns representing customer names or addresses, while each row contains data for one individual customer.



These tables can be linked or related using keys. Each row in a table is identified using a unique key, called a primary key. This primary key can be added to another table, becoming a foreign key. The primary/foreign key relationship forms the basis of the way relational databases work.
In our example, if we have a table representing product orders, one of the columns might contain customer information. Here, we can import a primary key that links to a row with the information for a specific customer.



This way, we can reference the data or duplicate data from the customer information table. It also means that these two tables are now related.

Since relational databases use tables of rows and columns, they display data more simply than some other database types, making them easier to use.

This tabular structure shifts the focus to handling data, which allows faster performance and the use of complex, high-level queries.

Finally, relational databases make it easy to scale data by simply adding rows, columns, or entire tables without changing the overall database structure.

Relational Database Features

The main features of a relational database are:

- **ACID compliant**. The database retrains integrity while performing transactions.
- **Range of data types.** Provides the capability to store any data as well as carry out complex queries.
- **Collaborative**. Multiple users are able to access the database and work on the same project.
- **Secure**. Access is limited or restricted through user permissions.
- **Stable.** Relational databases are well-understood and documented.

What are Relational Databases Used For?

Relational databases are the most implemented database type. There are many use cases, some of which include:

- **Online transaction systems.** The database supports many users as well as frequent queries needed in online transactions.
- **IoT**. Relational databases are lightweight and have the processing power needed for edge computing.
- **Data warehouses.** The critical component of the data warehouse architecture is storage. Relational databases are easily integrated and optimized for massive queries from multiple sources.

Most Popular Relational Databases

There are countless commercial as well as open-source databases. The top ten most popular relational databases are:

1. Oracle

2. MySQL

3. Microsoft SQL Server

4. PostgreSQL

5. IBM Db2

6. SQLite

7. Microsoft Access

8. MariaDB

9. Hive

10. Microsoft Azure SQL Database

Relational Database Advantages and Disadvantages

Like any other database model, there are advantages and disadvantages to using relational databases:

Advantages

Since relational databases use tables of rows and columns, they display data more simply than some other database types, making them easier to use.

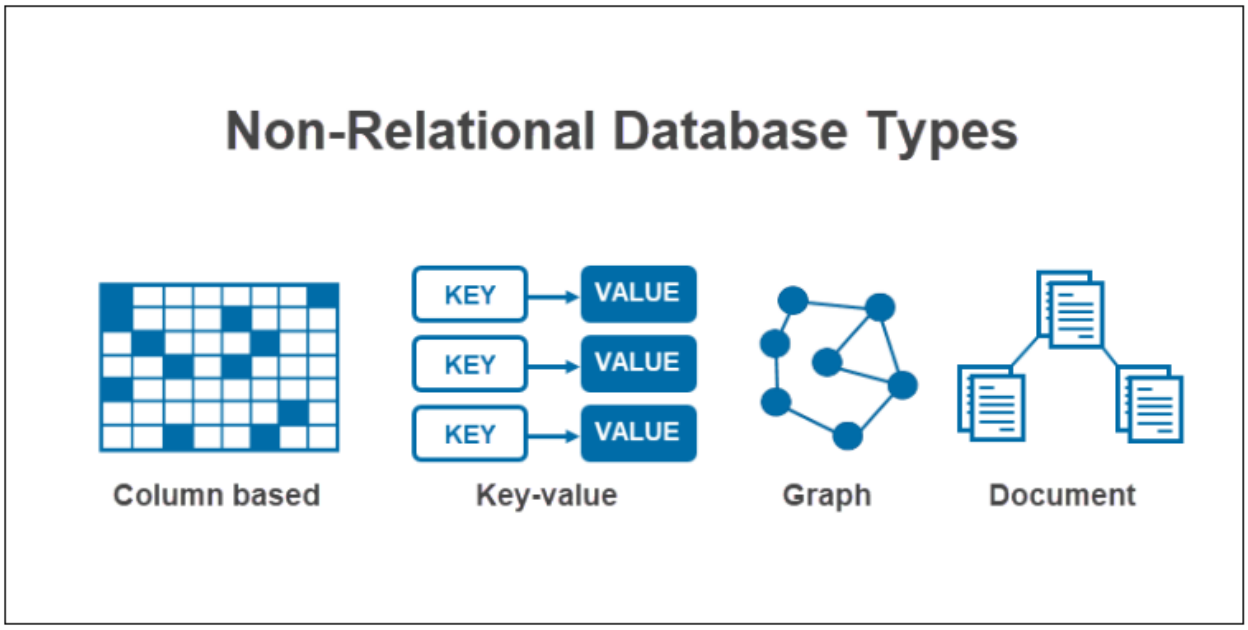
This tabular structure shifts the focus to handling data, which allows faster performance and the use of complex, high-level queries.

Finally, relational databases make it easy to scale data by simply adding rows, columns, or entire tables without changing the overall database structure.

Disadvantages

There are limits to how well relational databases can scale. In terms of sheer size, some databases have fixed limits on column lengths. If your database is built on a single dedicated server, scaling requires buying more server space, proving expensive in the long run.

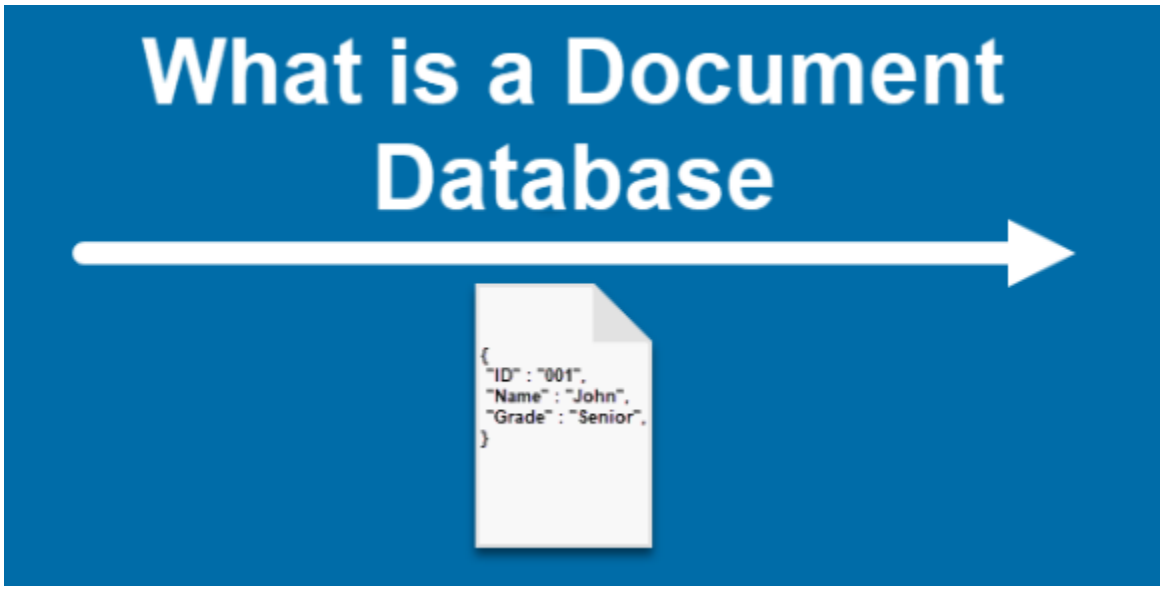
Also, constantly adding new elements to a database can make it so complex it becomes difficult to form relations between new pieces of data. Complicated data relations also slow down querying and negatively affect performance.



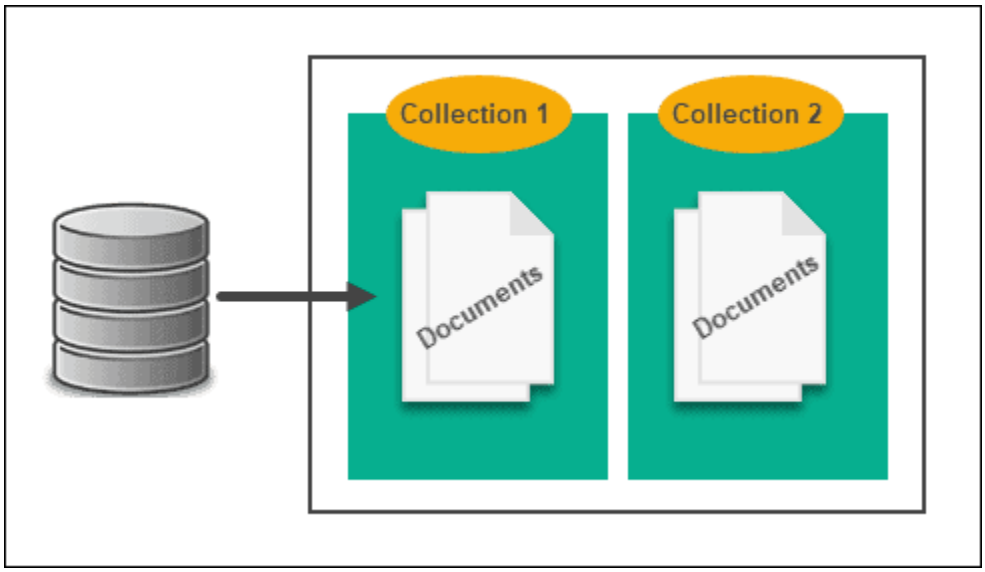
The 4 NoSQL database types are:

- Document
- Key-value
- Column based
- Graph

DOCUMENT DATABASE



A document database is a type of NoSQL database that consists of sets of key-value pairs stored into a document. These documents are basic units of data which you can also group into collections (databases) based on their functionality.



Each document consists of a number of key-value pairs. Here is an example of a document that consists of 4 key value pairs:

```
{
"ID" : "001",
"Book" : "Java: The Complete Reference",
"Genre" : "Reference work",
"Author" : "Herbert Schildt",
}
```

Using JSON enables app developers to store and query data in the same document-model format that they use to organize their app's code. The object model can be converted into other formats, such as JSON, BSON and XML.

Being a NoSQL database, you can easily store data without implementing a schema. You can transfer the object model directly into a document using several different formats. The most commonly used are JSON, BSON, and XML.

Relational Vs Document Database

Relational database management systems (RDBMS) rely on Structured Query Language (SQL). NoSQL doesn't.

A RDBMS is focused on creating relationships between files to store and read data. Document databases are focused on the data itself and relationships are represented with nested data.

Key comparisons between relational and document databases:

RDBMS	Document Database System
● Structured around the concept of relationships.	● Focused on data rather than relationships.
● Organizes data into tuples (or rows).	● Documents have properties without theoretical definitions, instead of rows.
● Defines data (forms relationships) via constraints and foreign keys (e.g., a child table references to the master table via its ID).	● No DDL language for defining schemas.
● Uses DDL (Data Definition Language) to create relationships.	● Relationships represented via nested data, not foreign keys (any document may contain others nested inside of it, leading to an N:1 or 1:N relationship between the two document entities).

• Offers extreme consistency, critical for some use cases such as daily banking.	• Offers eventual consistency with a period of inconsistency.

Features of Document Databases

Document databases provide fast queries, a structure well suited for handling big data, flexible indexing and a simplified method of maintaining the database. It's efficient for web apps and has been fully integrated by large-scale IT companies like Amazon.

Although SQL databases have great stability and vertical power, they struggle with super-sized databases. Use cases that require immediate access to data, such as healthcare apps, are a better fit for document databases. Document databases make it easy to query data with the same document-model used to code the application.

Document Databases Use Cases

General Use Cases	
User profiles	Extracting real-time big data
Book databases	Data of varying structures
Content management	Catalogs

Patients' data	

Book Database

Both relational and NoSQL document systems are used to form a book database, although in different ways.

The relational approach would represent the relationship between books and authors via tables with IDs – an *Author* table and a *Books* table. It forces each author to have at least one entry in the Books table by disallowing null values.

By comparison, the document model lets you nest. It shows relationships more naturally and simply by ensuring that each author document has a property called *Books*, with an array of related book documents in the property. When you search for an author, the entire book collection appears.

Content Management

Developers use document databases to create video streaming platforms, blogs and similar services. Each file is stored as a single document and the database is easier to maintain as the service evolves over time. Significant data modifications, such as data model changes, require no downtime as no schema update is necessary.

Catalogs

Document databases are much more efficient than relational databases when it comes to storing and reading catalog files. Catalogs may have thousands of attributes stored and document databases provide fast reading times. In document databases, attributes related to a single product are stored in a single document. Modifying one product's attributes does not affect other documents.

Document Database Advantages and Disadvantages

Below are some key advantages and disadvantages of document databases:

Document Database Advantages	Document Database Disadvantages
Schema-less	Consistency-Check Limitations
Faster creation and care	Atomicity weaknesses
No foreign keys	Security
Open formats	
Built-in versioning	

Advantages

- Schema-less. There are no restrictions in the format and structure of data storage. This is good for retaining existing data at massive volumes and different structural states, especially in a continuously transforming system.
- Faster creation and care. Minimal maintenance is required once you create the document, which can be as simple as adding your complex object once.
- No foreign keys. With the absence of this relationship dynamic, documents can be independent of one another.
- Open formats. A clean build process that uses XML, JSON and other derivatives to describe documents.

- Built-in versioning. As your documents grow in size they can also grow in complexity. Versioning decreases conflicts.

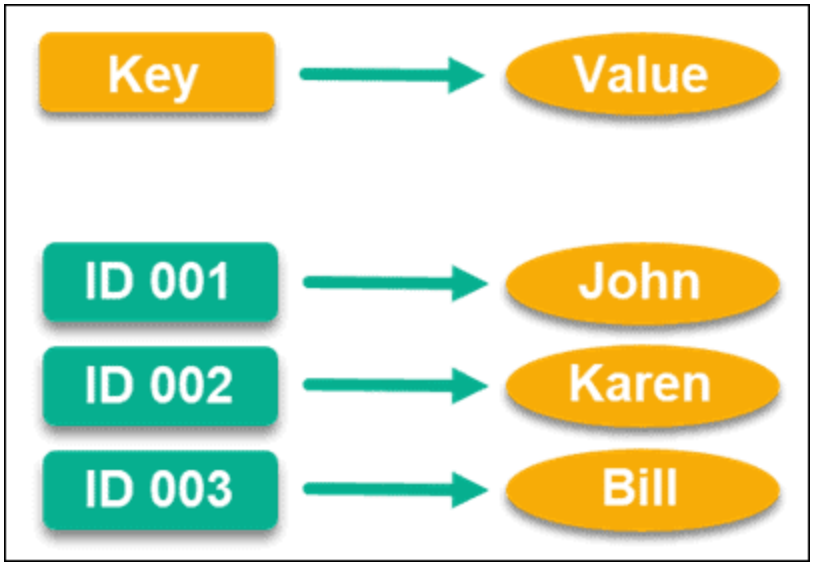
Disadvantages

- Consistency-Check Limitations. In the book database use case example above, it would be possible to search for books from a non-existent author. You could search the book collection and find documents that are not connected to an author collection.
  Each listing may also duplicate author information for each book. These inconsistencies aren't significant in some contexts, but at upper-tier standards of RDB consistency audits, they seriously hamper database performance.
- Atomicity weaknesses. Relational systems also let you modify data from one place without the need for JOINs. All new reading queries will inherit changes made to your data via a single command (such as updating or deleting a row). For document databases, a change involving two collections will require you to run two separate queries (per collection). This breaks atomicity requirements.
- Security. Nearly half of web applications today actively leak sensitive data. Owners of NoSQL databases, therefore, need to pay careful attention to web app vulnerabilities.

KEY-VALUE DATABASE

Key-value databases are the simplest type of NoSQL database. Thanks to their simplicity, they are also the most scalable, allowing horizontal scaling of large amounts of data.

These NoSQL databases have a dictionary data structure that consists of a set of objects that represent fields of data. Each object is assigned a unique key. To retrieve data stored in a particular object, you need to use a specific key. In turn, you get the value (i.e. data) assigned to the key. This value can be a number, a string, or even another set of key-value pairs.



Unlike traditional relational databases, key-value databases do not require a predefined structure. They offer more flexibility when storing data and have faster performance. Without having to rely on placeholders, key-value databases are a lighter solution as they require fewer resources.

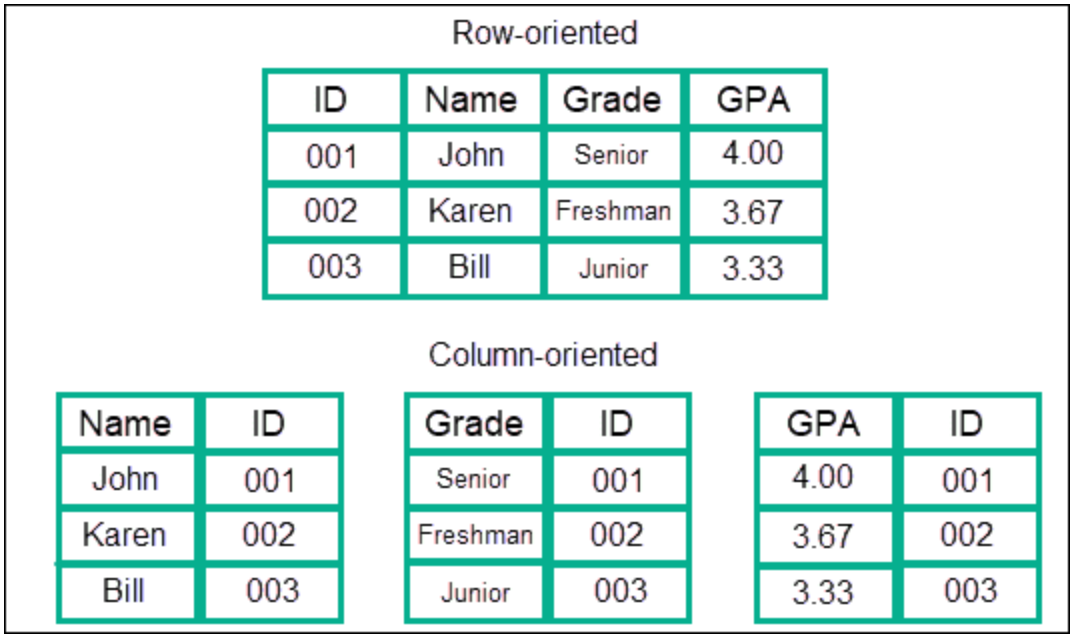
Such functionalities are suitable for large databases that deal with simple data. Therefore, they are commonly used for caching, storing, and managing user sessions, ad servicing, and recommendations.

Redis, Project Voldemort, and Riak are just some examples of key-value databases.

COLUMN DATABASE

Column-based databases focus on the efficiency of read operations. If you need to read several columns of multiple rows quickly, it makes sense to organize data in groups of columns (i.e., column families).

Column-base databases are another type of NoSQL database. In them, data is stored and grouped into separately stored columns instead of rows. Such databases organize information into columns that function similarly to tables in relational databases.

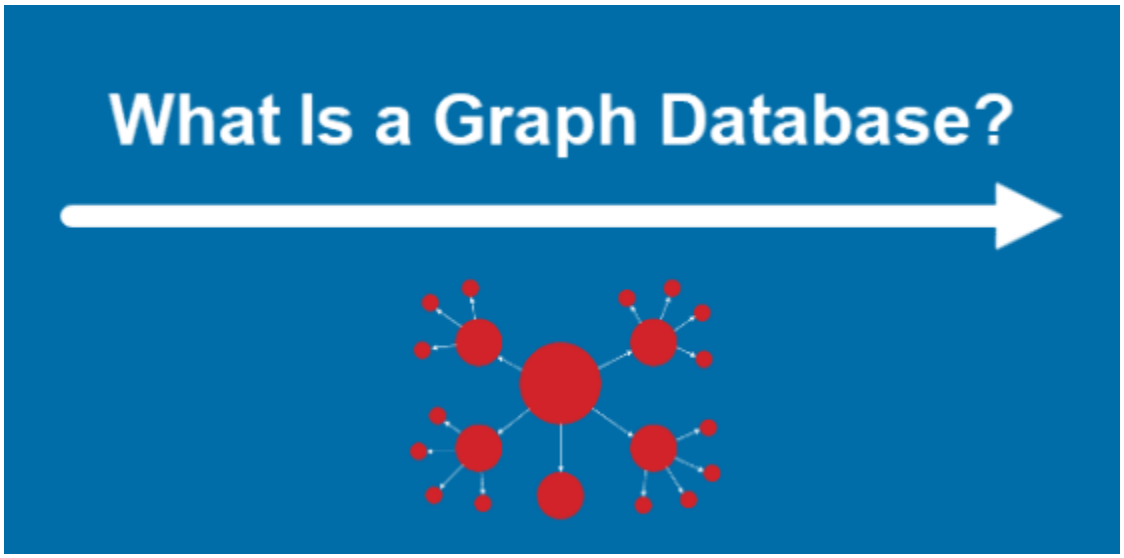


However, unlike traditional databases, wide-column databases are highly flexible. They have no predefined keys nor column names. Their schema-free characteristic allows variation of column names even within the same table, as well as adding columns in real-time.

The most significant benefit of having column-oriented databases is that you can store large amounts of data within a single column. This feature allows you to reduce disk resources and the time it takes to retrieve information from it. They are also excellent in situations when you have to spread data across multiple servers.

Examples of popular wide-column databases include Apache Cassandra, HBase, and CosmoDB.

GRAPH DATABASE



Graph databases are NoSQL systems created for exploring correlation within complexly interconnected entities. The structure addresses the limitations found in relational databases by putting a greater accent on the data relationship.

The graph database approach allows for more leisurely interconnection exploration, providing answers to complex questions about how data points relate to each other.

What Is a Graph Database?

A graph database is a NoSQL-type database system based on a topographical network structure. The idea stems from graph theory in mathematics, where graphs represent data sets using nodes, edges, and properties.

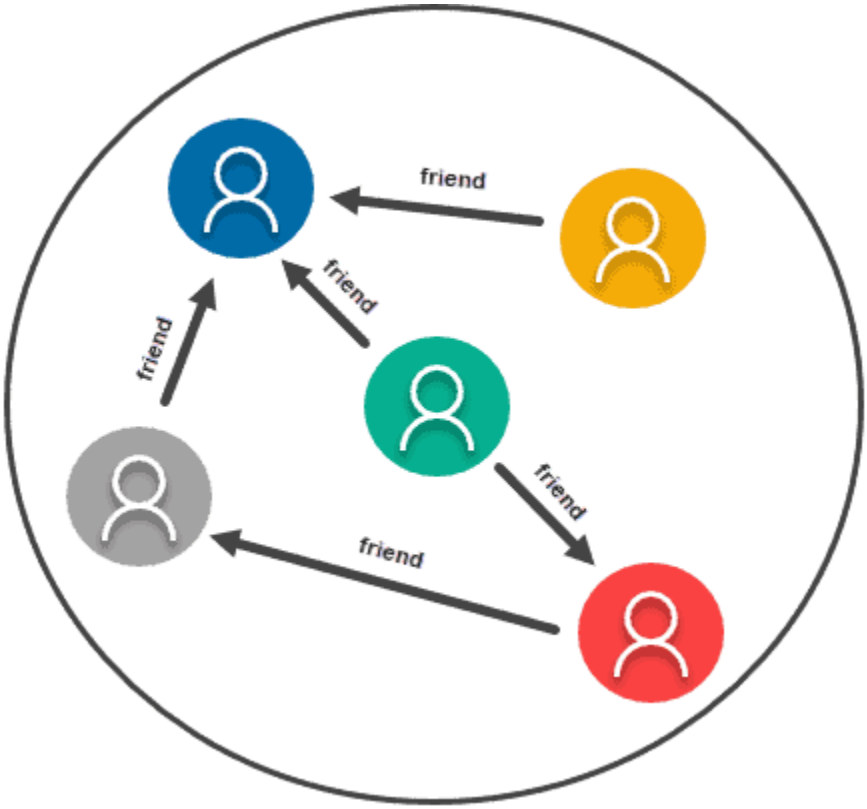
- **Nodes** or points are instances or entities of data which represent any object to be tracked, such as people, accounts, locations, etc.
- **Edges** or lines are the critical concepts in graph databases which represent relationships between nodes. The connections have a direction that is either unidirectional (one way) or bidirectional (two way).
- **Properties** represent descriptive information associated with nodes. In some cases, edges have properties as well.

Graph databases use flexible graphical representation to manage data.

These graphs consist of two elements:

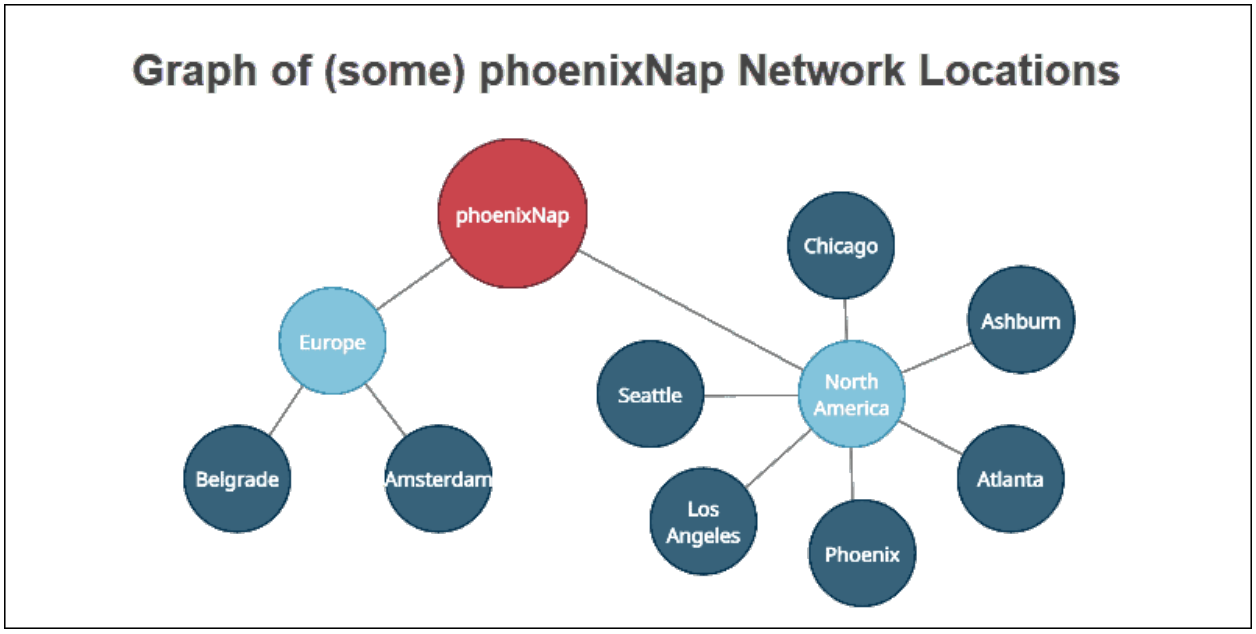
1. **Nodes (for storing data entities)**
2. **Edges (for storing the relationship between entities)**

These relationships between entities allow data in the store to be linked together directly and, in many cases, retrieved with one operation. Nodes and edges have defined properties, and by using these properties, you can query data easily.



Since this type of data-storing is quite specific, it is not a commonly used NoSQL database. However, there are certain use cases in which having graphical representations is the best solution. For instance, social networks often use graphs to store information about how their users are linked.

For example, analyze some of the network locations of phoenixNap:



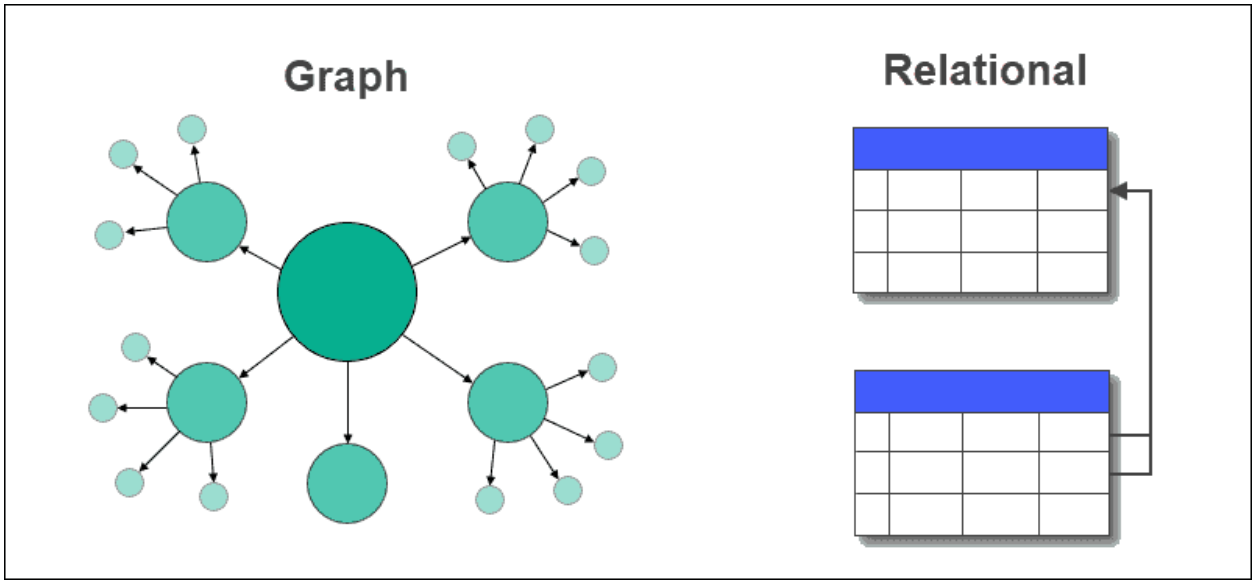
Nodes with descriptive properties form relationships represented by **edges.**

Graph databases provide a conceptual view of data more closely related to the real world. Modeling complex connections becomes easier since relationships between data points are given an equal value of importance as the data itself.

Graph Database vs. Relational Database

Graph databases are not meant to replace relational databases. As of now, relational databases are the industry standard. The most important aspect is to know what each database type has to offer.

Relational databases provide a structured approach to data, whereas graph databases are agile and focus on quick data relationship insight.



Both graph and relational databases have their domain. Use cases with complex relationships leverage the power of graph databases, outperforming traditional relational databases. Relational databases such as MySQL or PostgreSQL require careful planning when creating database models, whereas graphs have a much more naturalistic and fluid approach to data.

The following table outlines the critical differences between graph and relational databases:

Type	**Graph**	**Relational**
Format	Nodes and edges with properties	Tables with rows and columns
Relationships	Represented with edges between nodes	Created using foreign keys between tables
Flexibility	Flexible	Rigid

Complex queries	Quick and responsive	Requires complex joins
Use-case	Systems with highly connected relationships	Transaction focused systems with more straightforward relationships

How Do Graph Databases Work?

Graph databases work by treating data and relationships between data equally. Related nodes are physically connected, and the physical connection is also treated as a piece of data.

Modeling data in this way allows querying relationships in the same manner as querying the data itself. Instead of calculating and querying the connection steps, graph databases read the relationship from storage directly.

Graph databases are more closely related to other NoSQL data modeling techniques in terms of agility, performance, and flexibility. Like other NoSQL databases, graphs do not have schemas, which makes the model flexible and easy to alter along the way.

Graph Database Use Case Examples

There are many notable examples where graph databases outperform other database modeling techniques, some of which include:

- **Real-Time Recommendation Engines.** Real-time product and ecommerce recommendations provide a better user experience while maximizing profitability. Notable cases include Netflix, eBay, and Walmart.
- **Master Data Management.** Linking all company data to one location for a single point of reference provides data consistency and accuracy. Master data management is crucial for large-scale global companies.
- **GDPR and regulation compliances.** Graphs make tracking of data movement and security easier to manage. The databases reduce the potential of data breaches and provide better consistency when removing data, improving the overall trust with sensitive information.

- **Digital asset management.** The amount of digital content is massive and constantly increasing. Graph databases provide a scalable and straightforward database model to keep track of digital assets, such as documents, evaluations, contracts, etc.
- **Context-aware services.** Graphs help provide services related to actual-world characteristics. Whether it is natural disaster warnings, traffic updates, or product recommendations for a given location, graph databases offer a logical solution to real-life circumstances.
- **Fraud detection.** Finding suspicious patterns and uncovering fraudulent payment transactions is done in real-time using graph databases. Targeting and isolating parts of graphs provide quicker detection of deceptive behavior.
- **Semantic search.** Natural language processing is ambiguous. Semantic searches help provide meaning behind keywords for more relevant results, which is easier to map using graph databases.
- **Network management.** Networks are linked graphs in their essence. Graphs reduce the time needed to alert a network administrator about problems in a network.
- **Routing.** Information travels through a network by finding optimal paths makes graph databases the perfect choice for routing.

What Are the Well-Known Graph Databases?

Graph databases became more popular with the rise of big data and social media analytics. Many **multi-model databases** support graph modeling. However, there are numerous graph native databases available as well.

JanusGraph

JanusGraph is a distributed, open-source and scalable graph database system with a wide range of integration options catered to big data analytics. Some of the main features of JanusGraph include:

- Support for ACID transactions with the ability to bear thousands of concurrent users.
- Multiple options for storing the graph data, such as Cassandra or HBase.
- Complex search available by default as well as optional support for Elasticsearch.
- Full integration with Apache Spark for advanced data analytics.

- JanusGraph uses the graph transversal query language Gremlin, which is Turing complete.

Neo4j

Neo4j (**N**etwork **E**xploration and **O**ptimization **4 J**ava) is a graph database written in Java with native graph storage and processing. The main features of Neo4j are:

- The database is scalable through data partitioning into pieces known as shards.
- High availability provided through continuous backups and rolling upgrades.
- Multiple instances of databases are separable while remaining on one dedicated server, providing a high level of security.
- Neo4j uses the Cypher graph query language, which is programmer friendly.

DGraph

DGraph (**D**istributed **graph**) is an open-source distributed graph database system designed with scalability in mind. Some exciting features of DGraph include:

- Horizontal scalability for running in production with ACID transactions.
- DGraph is an open-source system with support for many open standards.
- The query language is GraphQL, which is designed for APIs.

DataStax Enterprise Graph

The **DataStax Enterprise Graph** is a distributed graph database based on Cassandra and optimized for enterprises. Features include:

- DataStax provides continuous availability for enterprise needs.
- The database integrates with offline Apache Spark seamlessly.
- Real-time search and analytics are fully integrated.
- Scalability available through multiple data centers.
- It supports Gremlin as well as CQL for querying.

Graph Database Advantages and Disadvantages

Every database type comes with strengths and weaknesses. The most important aspect is to know the differences as well as available options for specific problems. Graph databases are a growing technology with different objectives than other database types.

Advantages

Some advantages of graph databases include:

- The structures are agile and flexible.
- The representation of relationships between entities is explicit.
- Queries output real-time results. The speed depends on the number of relationships.

Disadvantages

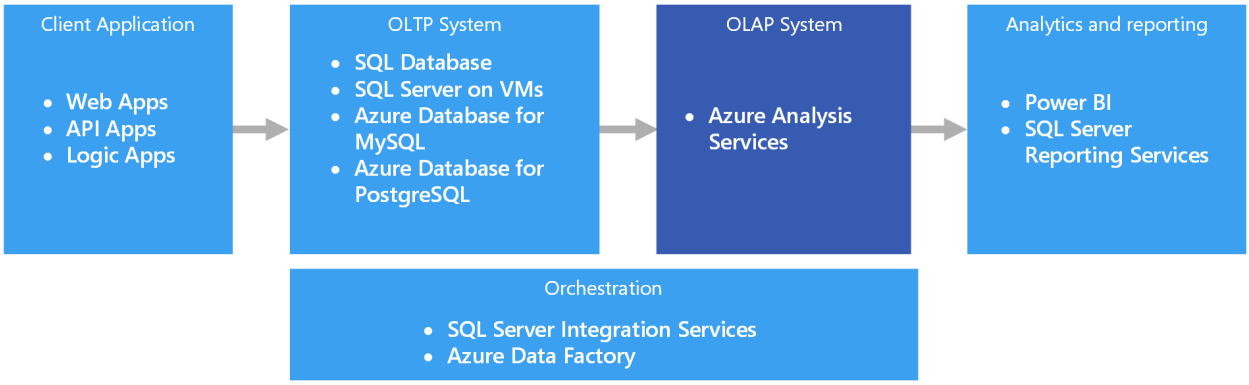
The general disadvantages of graph databases are:

- There is no standardized query language. The language depends on the platform used.
- Graphs are inappropriate for transactional-based systems.
- The user-base is small, making it hard to find support when running into a problem.

ANALYTICAL OLAP

Online analytical processing (OLAP) is a technology that organizes large business databases and supports complex analysis. It can be used to perform complex analytical queries without negatively affecting transactional systems.

The databases that a business uses to store all its transactions and records are called online transaction processing (OLTP) databases. These databases usually have records that are entered one at a time. Often they contain a great deal of information that is valuable to the organization. The databases that are used for OLTP, however, were not designed for analysis. Therefore, retrieving answers from these databases is costly in terms of time and effort. OLAP systems were designed to help extract this business intelligence information from the data in a highly performant way. This is because OLAP databases are optimized for heavy read, low write workloads.



OLTP and **OLAP** are online processing systems that help turn data into information. **OLTP** deals with **data transactions**, while **OLAP** deals with **data analytics**. Although there are differences, the main idea is to use the two processes to form a stable data warehouse architecture.

OLTP vs. OLAP: Definitions

The first step in comprehending the main difference between the OLTP and OLAP systems is to know how to define each.

OLTP (Online Transaction Processing)

OLTP is short for **Online Transaction Processing**. The system provides data to a dedicated storage server directly from the source. The main characteristics of OLTP are:

- **Frequent query processing**. Inserting, updating, and deleting data are everyday tasks in an OLTP database.
- **Fast transactions.** The system constantly deals with short and frequent transactions to stay updated with the most current information.
- **Data integrity.** In case of any failures, rollback segments are crucial to keeping data integrity as well as consistency. The stability of information flow is possible due to database normalization up to at least the third normal form (3NF).

Overall, the OLTP system design provides immediate response to simple business processes and user requests through a relational database.

OLAP (Online Analytical Processing)

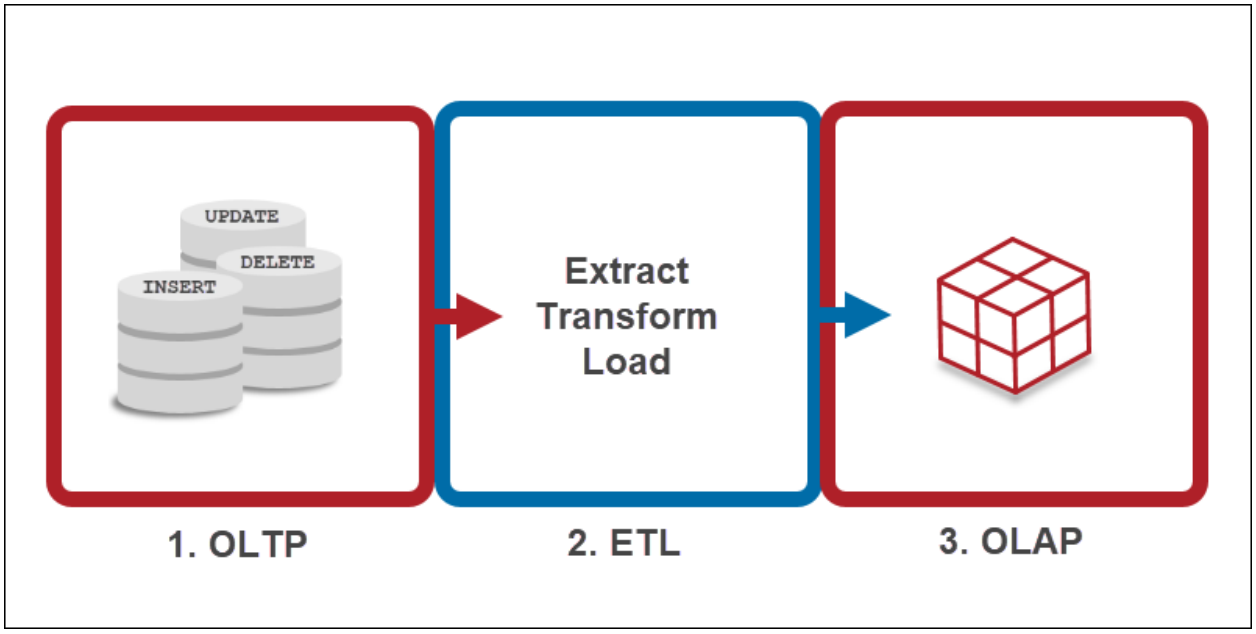
OLAP is short for **Online Analytical Processing**. The method takes data collected by an OLTP system and prepares it for analytical purposes. The main characteristics of an OLAP system are:

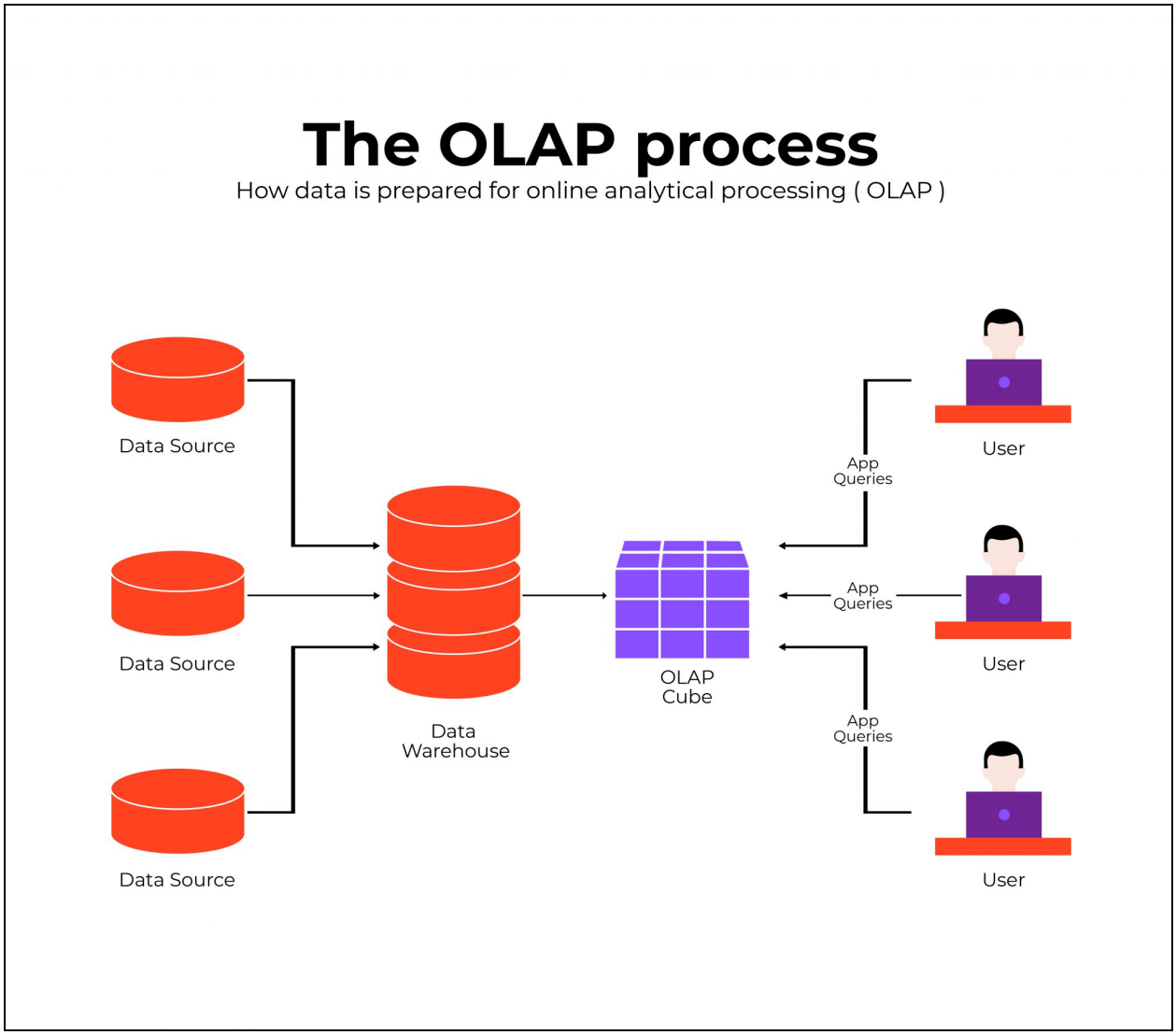
- **Smaller query volume.** Selecting multidimensional data is a common task in an OLAP database.
- **Complex transactions.** The system handles historical data and tackles analytical tasks in large volumes. The emphasis is on executing complex queries quickly for decision-making processes.
- **Query speed.** Database denormalization techniques help improve the query speed with OLAP databases. Although information retrieval is fast, data inconsistencies are present.

The OLAP system provides quick responses to complex and multidimensional workloads typically needed in a data warehouse.

OLTP vs. OLAP: Comparison

OLTP and OLAP are different in terms of functionality. The OLAP database systems became more popular with the rise of big data and analytics. The two systems work best when connected through the ETL (extract, transform, load) layer.





The table below outlines the main differences between the two processing techniques:

	OLTP	**OLAP**
Stands for	Online Transactional Processing	Online Analytical Processing

Functionality	Modifies and writes data often	Queries data, rarely writes
Main Feature	Low latency	High throughput
Queries	`INSERT`, `UPDATE`, `DELETE`	`SELECT`
Query Complexity	Simple and standardized	Complex and specialized
Normalization	Normalized	Unnormalized or denormalized
Database Architecture	Traditional	Data warehouse
Design	Industry oriented	Subject oriented
Integrity	Frequently modified and maintained	Not often changed or maintained
Data Redundancy	Low	High
Availability	High availability	Low availability
Storage Size	Small if data is archived	Large database servers

Number of Users	Thousands	Hundreds
Productivity	Short term and daily goals	Long term goals
Performance metric	Transaction throughput	Query throughput
Response time	Milliseconds	Seconds to minutes
Used for	Basic business tasks in high volumes	Planning, analytical tasks, decision making
Used by	Clerks, administrators, and data critical sectors	Data scientists, marketing, and decision-making sectors
Audience	Market-oriented information	Customer-oriented information

OLTP Use Cases

OLTP systems are in nearly every consumer-facing system. Some of the everyday use cases for transactional processing are:

- **ATM and Online Banking.** Daily financial withdrawals and payments represent simple everyday transactions supported by OLTP systems.
- **Payment Processing**. Both online and in-store payments are transactional processes, whether it is a debit or credit card.

- **Online Booking.** Any reservation, ticketing, and booking system require OLTP methods and specifications.
- **Recordkeeping.** Whether the records are medical, educational, inventory control, or a customer service ticketing system, record keeping is a process that needs fast-paced management.

OLAP Use Cases

An OLAP system is found in every branch of business that benefits from data analysis. Frequently, analytical processing finds use in:

- **Trend Analysis**. OLAP systems assist in decision-making with statistical analysis of trends in many sectors, from healthcare to retail.
- **Customer Behavior.** Different dimensions of customer information, such as geographic or demographic data, helps determine customer behavior for ecommerce industries.
- **Agriculture.** A recent and most exciting application is in the agriculture sector. Massive amounts of information processed with edge computing help generate reports for rural businesses.

Advantages and Disadvantages of OLTP

OLTP is system-oriented towards a high number of simple transactions with immediate responses. The transactional data processing technique has certain benefits and drawbacks.

Advantages

Some advantages of using OLTP are:

- **Concurrency.** A high volume of transactions from numerous users requires a high level of concurrency.
- **Atomicity.** Either a whole transaction occurs, or nothing happens. The system is immune to partial updates and information loss.
- **Speed.** All the occurring transactions are simple. Constant updates require sub-second response times.

Disadvantages

The disadvantages of OLTP include:

- **Downtime.** Any downtime causes a bottleneck in the high volume of requests. The systems must use high availability solutions.
- **Security.** When dealing with any data regarding people, safety is of the utmost priority. OLTP requires high levels of security, which is hard to manage with the massive number of transactions.
- **Request volume.** The sheer number of requests is overwhelming. The amount of raw data requires a team of data experts to find actionable information.

Advantages and Disadvantages of OLAP

OLAP focuses on data discovery processes and multidimensionality. The analytical approach to database analysis comes with advantages as well as disadvantages.

Advantages

The overall advantages to using an OLAP system are:

- **Comprehensive.** Complex queries on multidimensional data provide a broad overview of information from various databases.
- **Decision-making support**. With the help of star and snowflake schemas, the OLAP system provides the flexibility needed for decision support systems.
- **Flat learning curve.** The end-users of OLAP-based systems need little to no technical training.

Disadvantages

Some of the weaknesses in OLAP systems are:

- **Data redundancy.** High levels of data redundancy are present due to denormalization.
- **Storage scalability**. The system requires a scalable storage solution as the information system grows.
- **Computation capabilities.** Because non-technical professionals use OLAP systems, the computation resources lack power. Often, third-party software and tools are needed to perform complex computations.