Types of Databases

Alice M. Villarubia

3-BSCS-A

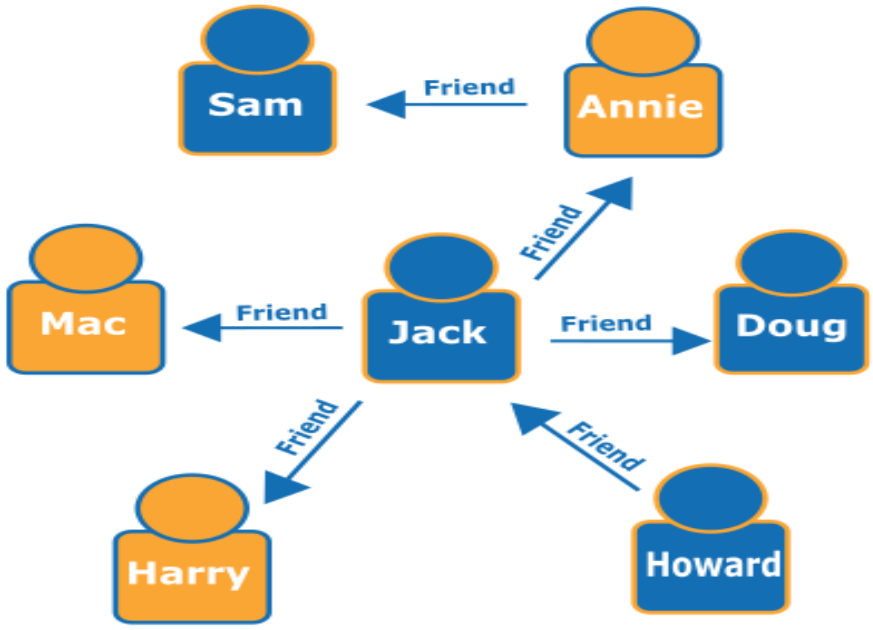
Graph database

Graph databases are databases that focus equally on the data and the connections between them. In this database, data is not constricted to predefined models. Most other databases can find connections between data when you run a search. With a graph database, these connections are stored inside the database right alongside the original data. This makes for a more efficient and faster database when your primary goal is to manage the connections between your data.

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

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

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



https://d1.awsstatic.com/diagrams/foaf-graph.e5e42865e0ee97a2972f9014d28f525ef68a981b.png

Use cases

Fraud detection

Graph databases are capable of sophisticated fraud prevention. With graph databases, you can use relationships to process financial and purchase transactions in near-real time. With fast graph queries, you are able to detect that, for example, a potential purchaser is using the same email address and credit card as included in a known fraud case. Graph databases can also help you easily detect relationship patterns such as multiple people associated with a personal email address, or multiple people sharing the same IP address but residing in different physical addresses.

Recommendation engines

Graph databases are a good choice for recommendation applications. With graph databases, you can store in a graph relationship between information categories such as customer interests, friends, and purchase history. You can use a highly available graph database to make product recommendations to a user based on which products are purchased by others who follow the same sport and have similar purchase history. Or, you can identify people who have a friend in common but don't yet know each other, and then make a friendship recommendation.

Source: https://aws.amazon.com/nosql/graph/

Relational database

Relational databases are the other major type of database, opposite of NoSQL. With a relational database, information is stored structured about other data. A good representation of a relational database would be the connection between a person shopping online and their shopping cart. Relational databases are often preferred when you are concerned about the integrity of your data, or when you're not particularly focused on scalability.

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. This data can be accessed in many different ways without reorganizing the database tables themselves.

Important Aspects of Relational Databases

SQL

SQL or Structured Query Language is the primary interface used to communicate with Relational Databases. SQL became a standard of the American National Standards Institute (ANSI) in 1986. The standard ANSI SQL is supported by all popular relational database engines, and some of these engines also have extension to ANSI SQL to support functionality which is specific to that engine. SQL is used to add, update or delete rows of data, retrieving subsets of data for transaction processing and analytics applications, and to manage all aspects of the database.

Data Integrity

Data integrity is the overall completeness, accuracy and consistency of data. Relational databases use a set of constraints to enforce data integrity in the database. These include primary Keys, Foreign Keys, ‘Not NULL’ constraint, ‘Unique’ constraint, ‘Default’ constraint and ‘Check’ constraints. These integrity constraints help enforce business rules on data in the tables to ensure the accuracy and reliability of the data. In addition to these, most relation databases also allow custom code to be embedded in triggers that execute based on an action on the database.

Transactions

A database transaction is one or more SQL statements that are executed as a sequence of operations that form a single logical unit of work. Transactions provide an "all-or-nothing" proposition, meaning that the entire transaction must complete as a single unit and be written to the database or none of the individual components of the transaction should go through. In the relation database terminology, a transaction results in a COMMIT or a ROLLBACK. Each transaction is treated in a coherent and reliable way independent of other transactions.

ACID Compliance

All database transactions must be ACID compliant or be Atomic, Consistent, Isolated and Durable to ensure data integrity.

Atomicity requires that either transaction as a whole be successfully executed or if a part of the transaction fails, then the entire transaction be invalidated. Consistency mandates the data written to the database as part of the transaction must adhere to all defined rules, and restrictions including constraints, cascades, and triggers. Isolation is critical to achieving concurrency control and makes sure each transaction is independent unto itself. Durability requires that all of the changes made to the database be permanent once a transaction is successfully completed.

Source: https://aws.amazon.com/relational-database/

Document databases

A document database is a type of nonrelational database that is designed to store and query data as JSON-like documents. Document databases make it easier for developers to store and query data in a database by using the same document-model format they use in their application code. The flexible, semistructured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs. The document model works well with use cases such as catalogs, user profiles, and content management systems where each document is unique and evolves over time. Document databases enable flexible indexing, powerful ad hoc queries, and analytics over collections of documents.

Use cases

Content management

A document database is a great choice for content management applications such as blogs and video platforms. With a document database, each entity that the application tracks can be stored as a single document. The document database is more intuitive for a developer to update an application as the requirements evolve. In addition, if the data

model needs to change, only the affected documents need to be updated. No schema update is required and no database downtime is necessary to make the changes.

Catalogs

Document databases are efficient and effective for storing catalog information. For example, in an e-commerce application, different products usually have different numbers of attributes. Managing thousands of attributes in relational databases is inefficient, and the reading performance is affected. Using a document database, each product's attributes can be described in a single document for easy management and faster reading speed. Changing the attributes of one product won't affect others.

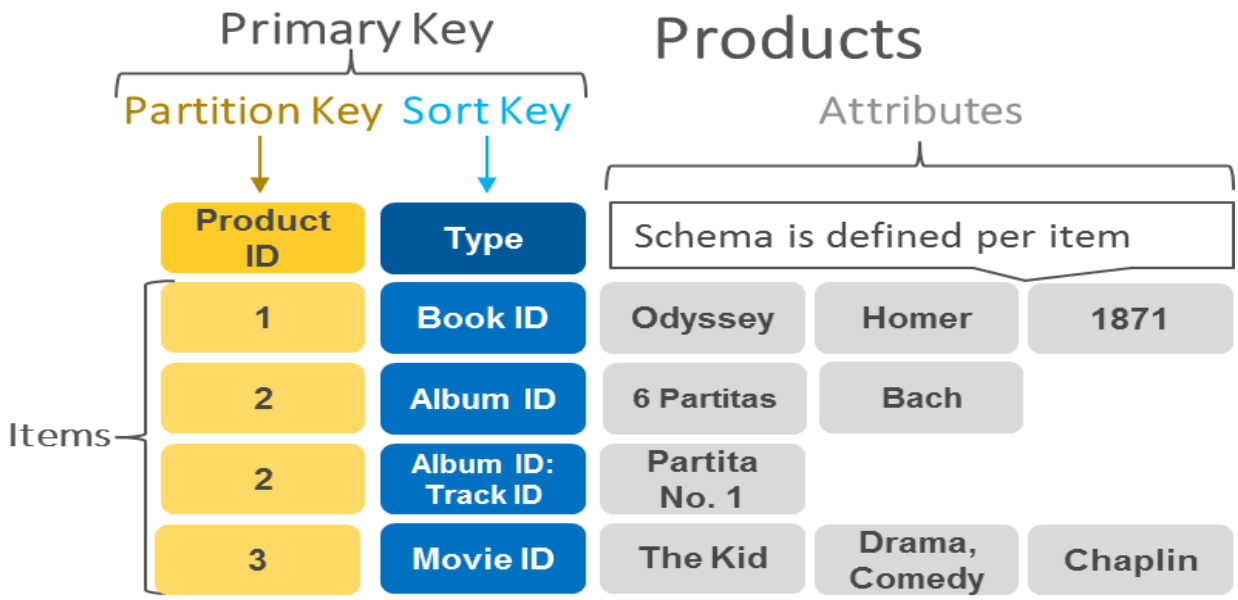
Source: https://aws.amazon.com/nosql/document/

Key-value databases

One of the simplest types of NoSQL databases, key-value databases save data as a group of key-value pairs made up of two data items each. They're also sometimes referred to as a key-value store. Key-value databases are highly scalable and can handle high volumes of traffic, making them ideal for processes such as session management for web applications, user sessions for massive multi-player online games, and online shopping carts.

A key-value database is a type of nonrelational database that uses a simple key-value method to store data. A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier. Both keys and values can be anything, ranging from simple objects to complex compound objects. Key-value databases are highly partitionable and allow horizontal scaling at scales that other types of databases cannot achieve. For example, Amazon DynamoDB allocates additional partitions to a table if an existing partition fills to capacity and more storage space is required.

The following diagram shows an example of data stored as key-value pairs in DynamoDB.



https://d1.awsstatic.com/product-marketing/DynamoDB/PartitionKey.8dd0530a7f6d66d101f31de30db515564f4cf28a.png

Use cases

Session store

A session-oriented application such as a web application starts a session when a user logs in and is active until the user logs out or the session times out. During this period, the application stores all session-related data either in the main memory or in a database. Session data may include user profile information, messages, personalized data and themes, recommendations, targeted promotions, and discounts. Each user session has a unique identifier. Session data is never queried by anything other than a primary key, so a fast key-value store is a better fit for session data. In general, key-value databases may provide smaller per-page overhead than relational databases.

Shopping cart

During the holiday shopping season, an e-commerce website may receive billions of orders in seconds. Key-value databases can handle the scaling of large amounts of data and extremely high volumes of state changes while servicing millions of simultaneous users through distributed processing and storage. Key-value databases also have built-in redundancy, which can handle the loss of storage nodes.

Source: https://aws.amazon.com/nosql/key-value/

Analytic Database

An analytic database, also called an analytical database, is a read-only system that stores historical data on business metrics such as sales performance and inventory levels. Business analysts, corporate executives and other workers run queries and reports against an analytic database. The information is regularly updated to include recent transaction data from an organization's operational systems.

An analytic database is specifically designed to support business intelligence (BI) and analytic applications, typically as part of a data warehouse or data mart. This differentiates it from an operational, transactional or online transaction processing database, which is used for transaction processing, such as order entry and other business applications.

While databases that do transaction processing can also support data warehouses and BI applications, analytic database vendors claim their products offer performance and scalability advantages over conventional relational database software.

Components and types of analytic databases

Analytical databases store massive amounts of data that organizations use to gain insight into their business, customers and more. The data stored in analytical databases comes from sources such as enterprise resource planning (ERP), customer relationship management (CRM) and other business applications or proprietary data. Some analytical databases include a data warehouse, which is designed to be flexible for users who need to create specific reports and queries.

Two major components of analytical databases are the data model and the query language. The data model is the structure through which data is stored. Data models can be relational -- where data is stored in separate tables and joined together when necessary -- or object-oriented -- where data is stored in fields of objects and represented as objects in software.

A query language is a standardized and interpreted programming language for the retrieval of information from a database management system (DBMS) or database. The

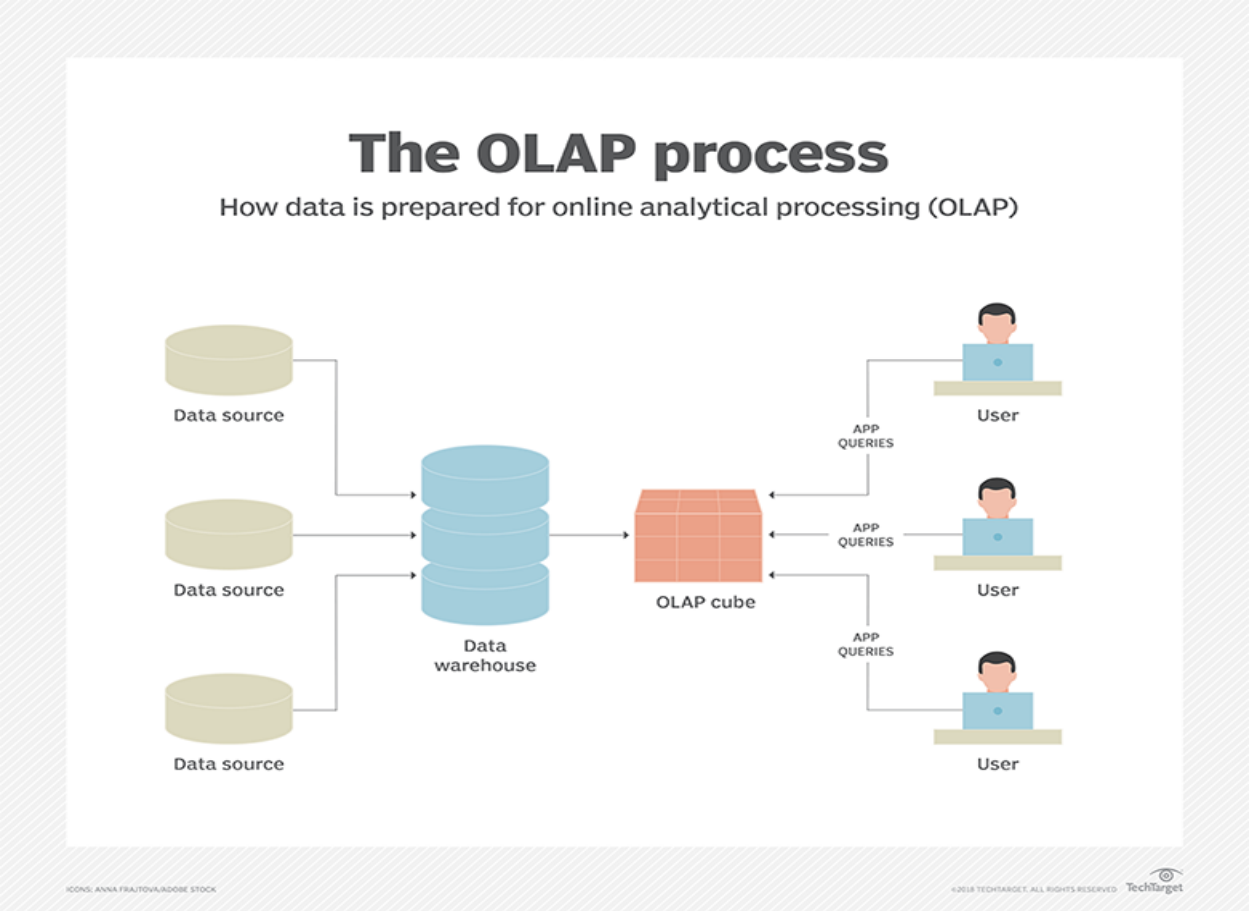
information is usually, but not always, in the form of a table or a set of tables. A query language enables users to retrieve information from a database without knowing the internal structure of the database. The most well-known query language is Structured Query Language (SQL). It is used to access and manipulate data in relational databases. SQL is a declarative programming language -- as opposed to a procedural one -- meaning that its syntax is defined by the data it operates on rather than the steps to manipulate the data.

There are five main types of analytic databases:

- **Columnar databases.** A columnar database stores data in large contiguous blocks of memory called data columns. This differs from a row-oriented database, where data is stored in tables split across columns and rows. It is the use of columns that makes columnar databases well suited to analytical processing, and able to meet the requirements of data warehouses. For example, columnar databases provide a solution to the problem of data sprawl. As data volumes grow, the value of high-speed columnar database technology scales. Because columnar databases are designed to store, analyze and retrieve data by column, they can accelerate data loading and improve data analytics performance. The columnar architecture of this database also allows for efficient data compression and fast aggregate queries. Examples of columnar databases include Amazon Redshift, MariaDB and Apache Kudu.
- **Massively parallel processing (MPP) databases.** In an MPP database, data is stored on multiple servers rather than on a single server. Each server has its own set of data and its own computing power. MPP databases typically provide high availability and redundancy by using clusters of computers to serve multiple users. They also efficiently handle large amounts of data while providing fast access to that data. Examples of MPP databases include Vertica and Teradata.
- **In-memory databases.** In-memory databases are optimized to store data in random-access memory. This contrasts with traditional databases that rely on hard drives to store both data and program instructions. Because data is stored in

memory, in-memory databases provide a near-real-time response. Storing data in memory offers several potential benefits over traditional databases, including faster query processing, less I/O overhead and the ability to handle larger databases. Examples of in-memory databases include SAP HANA, Oracle TimesTen and Aerospike.

- **Online analytical processing (OLAP) databases.** An OLAP database is an online database that is used to create reports and analyses that are faster and more efficient than traditional databases. An OLAP database stores multidimensional *cubes* of aggregated data for analyzing information based on multiple data attributes. The data is stored in a structure designed to facilitate analysis of the data. Because OLAP databases are stored in a multidimensional format, they can handle a high volume of data better than a traditional database. Examples of database structures include star schemas and snowflake schemas. OLAP databases are used in several industries, including financial services and retail. Examples of OLAP databases include Snowflake and IBM Cognos.
- **Data warehouse appliances.** Data warehouse appliances are integrated hardware and software tools designed to optimize the performance of data warehouses. A data warehouse is an information repository large organizations use to centralize, store and index vast amounts of data from disparate sources. Data warehouse appliances simplify and expedite data warehouse implementation and management. They are commonly used in BI to track and analyze data from many sources, including ERP systems, CRM software and other databases. Numerous vendors offer data warehouse appliances, including IBM, Microsoft, Oracle and Teradata.



https://cdn.ttgtmedia.com/rms/onlineimages/crm-olap_desktop.png

Source: https://searchbusinessanalytics.techtarget.com/definition/analytic-database

Column-Family Database

NoSQL column family database is another aggregate oriented database.
In **NoSQL** column family database we have a single key which is also known as row key and within that, we can store multiple column families where each column family is a combination of columns that fit together.

Column family as a whole is effectively your aggregate. We use row key and column family name to address a column family.

It is, however, one of the most complicated aggregate databases but the gain we have in terms of retrieval time of aggregate rows. When we are taking these aggregates into the memory, instead of spreading across a lot of individual records we store the whole thing in one database in one go.

The database is designed in such a way that it clearly knows what the aggregate boundaries are. This is very useful when we run this database on the cluster.

As we know that aggregate binds the data together, hence different aggregates are spread across different nodes in the cluster.

Therefore, if somebody wants to retrieve the data, say about a particular order, then you need to go to one node in the cluster instead of shooting on all other nodes to pick up different rows and aggregate it.

Among the most popular column family **NoSQL databases** are Apache **HBase** and Cassandra.

Aggregate orientation is not always a good thing.

Let us consider the user needs the revenue details by product. He does not care about the revenue by orders.

Effectively he wants to change the aggregate structure from order aggregate line item to produce aggregate line items. Therefore, the product becomes the root of the aggregate.

In a relational database, it is straightforward. We just query few tables and make joins and the result is there on your screen. But when it comes to aggregate orientation database it is a pain.

We have to run different **MapReduce** jobs to rearrange your data into different aggregate forms and keep doing the incremental update on aggregated data in order to serve your business requirement, but this is very complicated.

Therefore, the aggregate oriented database has an advantage if most of the time you use the same aggregate to push data back and forth into the system. It is a disadvantage if you want to slice and dice data in different ways.

Source: https://www.rcvacademy.com/nosql-column-family-database/