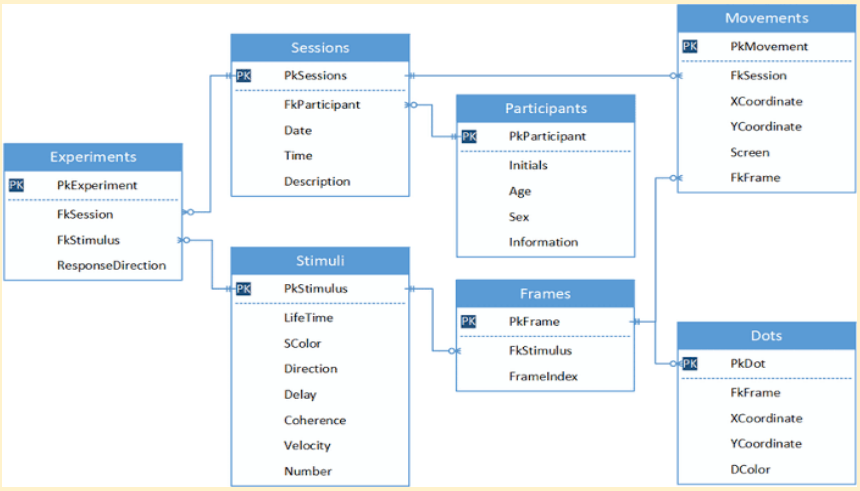
Applications Development
CS 312

Types of Databases

By: Diana Kate Magcanam Carpio
3BSCSA

1. Relational



A relational database is one that stores and provides access to data elements that are connected to one another. The relational model, which is a straightforward and intuitive means of presenting data in tables, is the foundation of relational databases. Each row in a table in a relational database is a record with a unique identifier called a key. Table columns store data attributes, and each record typically contains the value of each attribute, making it easy to determine the relationship between data points.

Any application can use the relational data model because it provides a standard way of viewing and querying data. For starters, developers have realized that the relational database model's greatest strength is the use of tables, which provide an intuitive, efficient, and flexible way to store and access embedded data.

As programmers began to use Advanced Query Language (SQL) to write data and query databases, the power of design metrics became obvious. SQL was widely used as a language for querying data for many years. SQL provides accurate arithmetic that optimizes all data queries because it is based on relational algebra. In contrast, some methods necessitate the explanation of specific requirements.

In 1970, EF Codd, a junior IBM programmer at the time, devised the relational database. Codd recommended transitioning from data storage to hierarchical structures or navigation to data organization in his work "Relational Data Model for Large Shared Databanks." Tables of data A table with columns and rows.

In a relational database, each table, also known as a relationship, includes one or more kinds of data in columns or attributes. Each row, also known as a record or tuple, includes a single instance of data (or key) for each of the columns' categories. Each table has a distinct primary key that identifies the data within it. Foreign keys, which are fields in one table that are linked to the primary key of another table, can be used to define the relationship between tables.

How it works is as follows:
The customer information table is the first, with each record containing the customer's name, address, and billing information. Each line of data has its own column, and the database assigns it a unique identity. The customer's name and contact details are not included in the second sales order table, which includes the selected product, quantity, size, and color, among other things.One thing these two tables have in common is the ID column. This common column, on the other hand, enables the relational database to interact between two tables. The order is subsequently sent to a database, which may access the sales order table, obtain the right product order information, and use the customer ID table to locate the customer's invoice and any shipments. Customer information is stored in a table. The store will be able to obtain the correct product, clientts will be able to place orders fast, and they will be able to pay the trade value.

A branch office manager, for example, might request a report on all customers who purchased products after a certain date. A financial services manager in the same company could get a report on accounts that needed to be paid using the same tables.

Users define the domain of potential values in a data column, as well as any constraints that may apply to that data value, while constructing a relational database. For example, a domain of possible customers could have up to ten possible customer names, but only three of these customer names could be specified in a single table.

Furthermore, relational databases are physically data independent. This refers to a system's ability to modify its internal schema without affecting other schemas or application programs. Alterations to the inner schema may include the following:

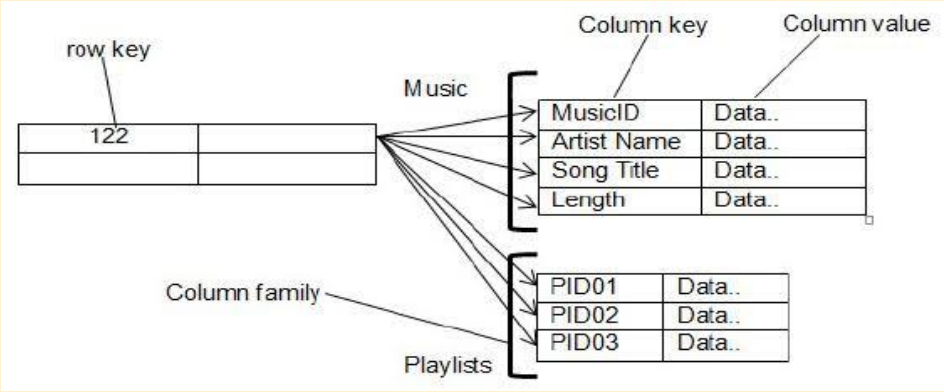
- the use of new storage devices;
- modifying indexes;
- changing from a specific access method to a different one;
- using different data structures; and
- using various storage structures or file organizations.

For a range of information requirements, companies of all types and sizes use a simple and effective relational model. Inventory tracking, e-commerce processing, and managing vast amounts of business-critical client data are all functions that relational databases are utilized for. Relational databases can handle all of the information needs that data points have and manage them in a consistent, safe, and rule-based manner.Since the 1970s, relational databases have been used. The advantages of the relational model have made it the most extensively used database model today.

Examples of relational database:Users can manage preset data relationships across various databases using standard relational databases. Microsoft SQL Server, Oracle Database, MySQL, and IBM DB2 are all examples of standard relational databases.

Database as a service, or cloud-based relational databases, are also popular because they allow businesses to outsource database maintenance, patching, and infrastructure support. Amazon Relational Database Service, Google Cloud SQL, IBM DB2 on Cloud, SQL Azure, and Oracle Cloud are examples of cloud relational databases.

2. Column - Family



A notion known as a keyspace is used to store databases in columns. In the relational paradigm, a keyspace is similar to a schema. All of the column families (similar to tables in the relational model) that contain rows and columns are contained in the keyspace.

Multiple rows make up a column family.
The number of columns in each row can differ from the number of columns in the other rows. Also, the columns in the first row do not have to match the columns in the second row (i.e. they can have different column names, data types, etc).

Each row is divided into columns. In contrast to a relational database, it does not span all rows. There is a name/value pair in each column, as well as a timestamp. Note that the timestamp in this example is in Unix/Epoch time.

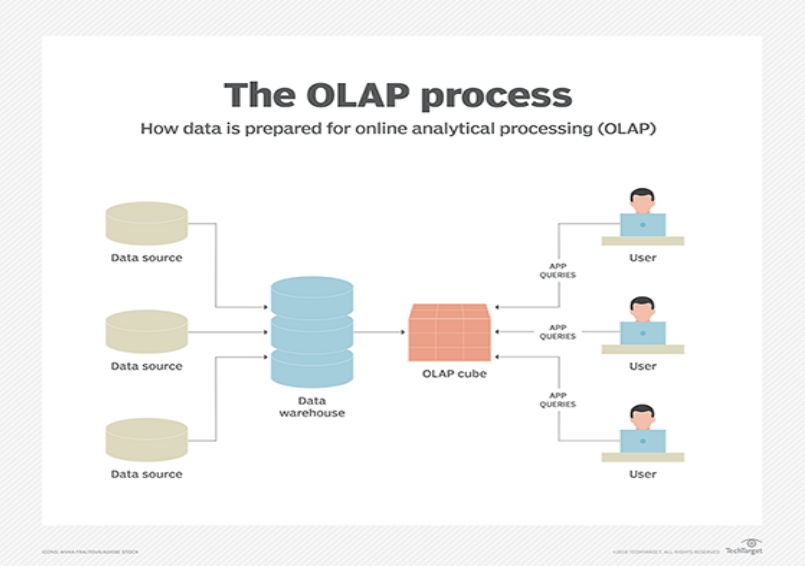
- Row is the most important key. Each row has its own unique key, which serves as a unique identifier.
- Column. A name, a value, and a timestamp appear in each column.
- Name. The name of the name/value pair is this.
- Value. This is the name/value pair's value.
- Timestamp. This information includes the date and time the data was entered. This can be used to find the most recent data version.

Some DBMSs extend the column family concept to provide additional functionality/storage capacity. Here are some of the advantages of Column Family Databases that make them a popular choice for organizations dealing with large amounts of data.

- Compression. Data compression and/or partitioning are both very efficient with column stores.
- Query aggregation. Columnar databases are particularly good at aggregating queries because of their structure (such as SUM, COUNT, AVG, etc).
- Scalability. Columnar databases have a lot of scalability. They're ideal for massively parallel processing (MPP), which entails distributing data over a huge cluster of devices - frequently thousands of them.
- Fast to load and query. Columnar stores can be loaded extremely fast. A billion-row table could be loaded within a few seconds. You can start querying and analyzing almost immediately.

Examples of Column Store DBMSs
Bigtable
Cassandra
HBase
Vertica
Druid
Accumulo
Hyper table

3. Analytical (OLAP)



OLAP (online analytical processing) is a computer approach that allows users to extract and query data quickly and selectively in order to examine it from many perspectives. Trend analysis, financial reporting, sales forecasting, budgeting, and other planning tasks are frequently aided by OLAP business intelligence queries.

For example, a user can ask for data to be examined in order to see a spreadsheet showing all of a company's beach ball products sold in Florida in July, compare revenue figures with those for the same products in September, and then compare other product sales in Florida during the same period.

Data is gathered from various sources and stored in data warehouses, where it is then cleansed and organized into data cubes. Data is categorized by dimensions (such as customers, geographic sales region, and period) in each OLAP cube, which are derived from dimensional tables in the data warehouses. Members (such as customer names, countries, and months) are then sorted hierarchically to populate dimensions. When compared to relational databases, OLAP cubes are frequently pre-summarized across dimensions.

Analysts can then use these multidimensional databases to perform five different types of OLAP analytical operations:

- Roll-up. This procedure, also known as consolidation or drill-up, summarizes data along a dimension.
- Drill-down. This allows analysts to drill down deeper into the data dimensions, such as charting sales growth for a product by drilling down from "period" to "years" and "months."
- Slice. This allows an analyst to exhibit a single level of data, such as "sales in 2017."

- Dice. This enables an analyst to assess data from several dimensions, such as "blue beach ball sales in Iowa in 2017."
- Pivot. By rotating the data axes of the cube, analysts can get a new perspective on the data.

The OLAP software then locates and displays the intersection of dimensions, such as all products sold in the Eastern region above a certain price during a given period. The result is the "measure," which is derived from data stored in fact tables in the data warehouse. Each OLAP cube has at least one and possibly hundreds of measures.

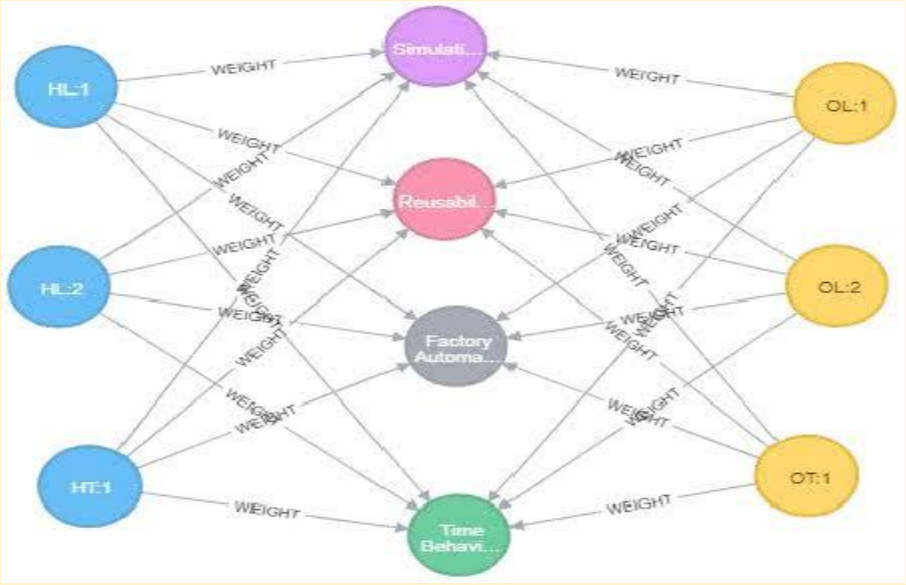
OLAP systems can be classified into a few different categories.
There are three types of OLAP (online analytical processing) systems:

- MOLAP (Multidimensional OLAP) is an OLAP that indexes into a multidimensional database directly.

- ROLAP stands for Relational OLAP, which is an OLAP that performs dynamic multidimensional analysis of data stored in a relational database.

- HOLAP stands for Hybrid OLAP,is the result of combining ROLAP and MOLAP. HOLAP was created to combine ROLAP's greater data capacity with MOLAP's superior processing capability.

4. Graph



The graph database was created with the intention of storing networks, or the connections between various items such as people, places they might visit, or goods they might use. Relational databases can find simple links between them, but when

asked to negotiate and evaluate complex relationships involving several parties, they become bogged down.

With the emergence of social networks, the graph database rose in popularity, but there's no reason to confine it to tracking people and their friendships. It may be used to find all of the books in Isaac Newton's library that cite another book. Alternatively, any chemical that reacts with table salt. Alternatively, any structure that may be reached by waiting at no more than two traffic signals.

Because mathematicians use the term "graph" to refer to several different constructs, most people are only familiar with the most well-known version: the line that plots the relationship between two variables such as time and money. Graph databases, on the other hand, are designed to store more ad hoc, non-continuous relationships.

When searching through the networks defined by these connections, the graph database really shines. They use specialized algorithms to compile the layers of relationships radiating out from a single entry.

Data is stored in graph databases using topographical data models. These databases employ graphs to connect specific data points (nodes) and generate relationships (edges) that can subsequently be retrieved by the user via queries. Nodes can represent consumers, companies, or any other data that a corporation wants to keep track of. The database creates edges to help the user understand the relationships between nodes. When businesses need to pull data and don't want to spend time structuring it into discrete relationships, they can use graph databases.

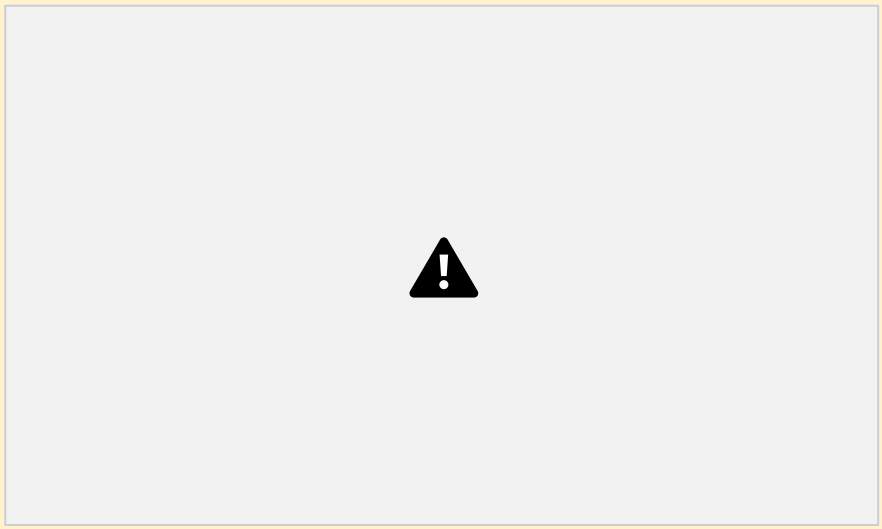
Complex inquiries can be used by large businesses to extract precise and in-depth information on their customers and users, as well as product tracking data.

Database administrators can develop usable models even with large amounts of data. An RDF database, a sort of graph database that focuses on retrieving triples, or information arranged in a subject-predicate-object connection, may be used by some companies. Document database tools, key-value store database tools, object-oriented database tools, and other database types are examples of similar databases. Free database software can be used by developers seeking for a cost-effective alternative.

Over the next decade, graph databases are likely to play a key role in areas as diverse as machine learning, Bayesian analysis, data science, and artificial intelligence, as well as assisting with enterprise data management and data transfer.

Improved data federation will be one of the most significant influences on this sort of database. When knowledge graphs are easily federated, one database will be able to recognize when it requires data that it does not have and will automatically retrieve that data from another knowledge graph. With this capability, federation is anticipated to assist developers in creating blockchains that employ relevant metadata to authenticate transactions in banking, finance, and other industries.

5. Key-Value



Data is kept in a "key-value" format and optimized for reading and writing in key value databases, also known as key value stores. To retrieve the related value with each key, a unique key or a set of unique keys are used to fetch the data. Simple data types like characters and numbers, as well as sophisticated objects, can be used as values.

A key-value database, also known as a key-value store, associates a value (which can range from a number or simple string to a complicated object) with a key, which is used to monitor the item. In its most basic form, a key-value store is similar to a dictionary/array/map object seen in most programming paradigms, but it is persistently saved and controlled by a Database Management System (DBMS).

Key-value databases use small, efficient index structures to locate a value by its key fast and consistently, making them excellent for systems that must discover and retrieve data in real time.Redis, for example, is a key-value database designed to track relatively simple data structures in a persistent database (primitive types, lists, heaps, and maps). Redis is able to present a very simple interface for querying and manipulating value types by only supporting a small number of them, and when configured properly, is capable of high performance.

A key-value database is defined as one that allows programs or program users to obtain data using keys, which are effectively names or identifiers that refer to a stored value.

- Finding and retrieving a value (if one exists) stored and associated with a particular key.
- Deletes the value stored and associated with a particular key (if one exists).
- Setting, updating, and changing the value associated with a given key (if one exists).

Modern applications would very certainly need more than this, but this is the absolute minimum for a key-value store.

When should you utilize it?
This may be volatile if your application must handle a large number of little continuous reads and writes. In-memory access is possible with key-value databases.
When saving fundamental information such as client information; when storing web pages with the URL as the key and the webpage as the value; when storing shopping cart contents, product categories, and e-commerce product details
For apps that don't need to be updated frequently or that can handle sophisticated queries.

In a simple form of a key-value fetch/update/remove, a key-value approach allows constructing an efficient and compact data structure to retrieve data.
Key-value stores are utilized in situations where programs need to retrieve valuesThis database architecture is a win for specific application workloads due to its compact structure and uncomplicated indexing and seeking through those indexes, similar to maps or dictionaries in computer languages.

A key-value store has a number of advantages over standard row-column-based databases. A key-value store can be particularly quick for read and write operations because to the basic data format that lends it its name. And, as we generate more data without traditional structures, key-value stores are a valuable asset in modern programming.

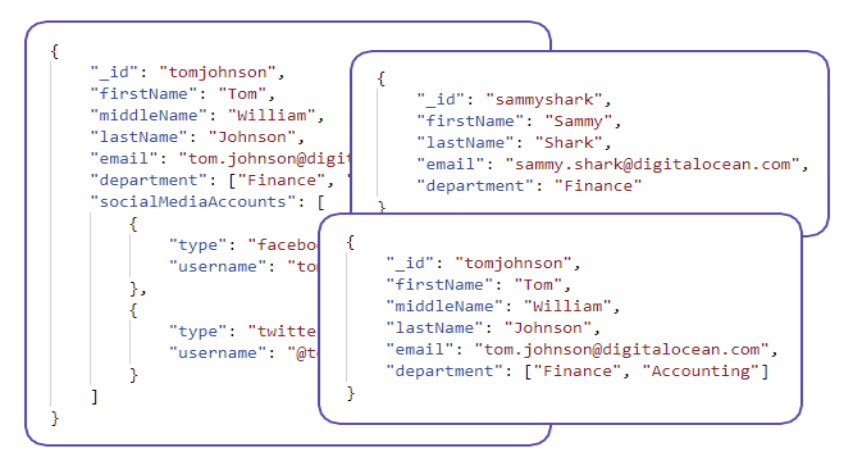
Furthermore, because key-value stores do not require placeholders for optional data such as "null," they may have lower storage requirements and grow virtually linearly with the number of nodes.

Examples of Popular Key-Value Databases

There are several different types of key-value database models to pick from, for example, some store data on an SSD, while others store on RAM. The truth is, some of the most popular and widely-used databases are key-value stores, and we rely on them on a daily basis in our day-to-day lives.

Amazon DynamoDB
Aerospike
Berkeley DB
Couchbase
Memcached
Risk
Redis

6. Document



In most cases, document databases store data as structured documents in the Extensible Markup Language (XML) or JSON formats. Document databases are, in some ways, the next step in the evolution of key-value stores. Document databases, on the one hand, use similar key-value stores and allow for nested key-value pairs. These databases outperform key-value storage in terms of query performance. Document databases, on the other hand, are thought to be an improvement over schema-less key-value stores since they use a self-described document format.

In contrast to rows or records in an RDBMS, document databases focus on storage and access methods optimized for documents. The data model is a set of document collections with key-value collections. The values in a document storage can be nested documents, lists, or scalar values. One of the significant differences between the advanced key-value stores we just discussed is nesting. The attribute names are dynamically created for each page during runtime, rather than being predefined in a global schema.In addition, unlike RDBMS tuples, a wide range of values is permitted. A document holds data in tree-like structures and requires that the data be saved in a database-friendly format. This storage format may theoretically be XML, JSON, Binary JSON (BSON), or anything else as long as the database can grasp the document's internal structure.

- Storage of dynamic data in unstructured, semi-structured, or organized formats.
- Ability to build and save persisting views from a base document for analysis.
- Capacity to store and process vast amounts of data.

Schema-free free—there are no restrictions on the structure and format of how the data must be stored—is one of the architectural features of document-oriented

databases. This adaptability allows an expanding system to incorporate more data while keeping the old data in the same structure.

- Document store—objects can be serialized and saved in a document, and there is no need to enforce and observe relational integrity.

- Ease of creation and maintenance—a simple document creation allows complicated items to be made only once, and once the document is created, minimal maintenance is required.

- No relationship enforcement—documents are independent of one another, and there is no need to care about foreign-key relationships while running queries. Concurrency effects and performance difficulties connected to concurrency aren't an issue here.

- Open formats—documents are specified using JSON, XML, or a derivative, ensuring an uniform and clean approach from the outset.

- Built-in versioning—with versions, documents can get huge and cluttered. Versioning is used by most solutions today to eliminate conflicts and maintain processing efficiencies.

Document databases express the data as files in JSON or XML formats. This allows the same document to be parsed for multiple contexts and the results scrapped and added to the next iteration of the database data.

To summarize, a document-oriented database is a NoSQL database that stores data as binary document files. Each document in this database is assigned a unique key, which might be a string, a URL, or a URI. Individual documents are located and pulled from the database using keys. A document store is another name for a document-oriented database.
The indexing options, as well as the API calls and query language, will change depending on the documents stored in the database. Depending on the content of the document, the manner it is arranged will also differ. Tags, metadata, and collections are commonly used to organize documents. One of the major benefits of adopting a document store is that if the data model needs to be modified in the future, just the relevant documents will need to be updated.

MongoDB, DynamoDB, and CosmosDB are examples of popular document-oriented databases.