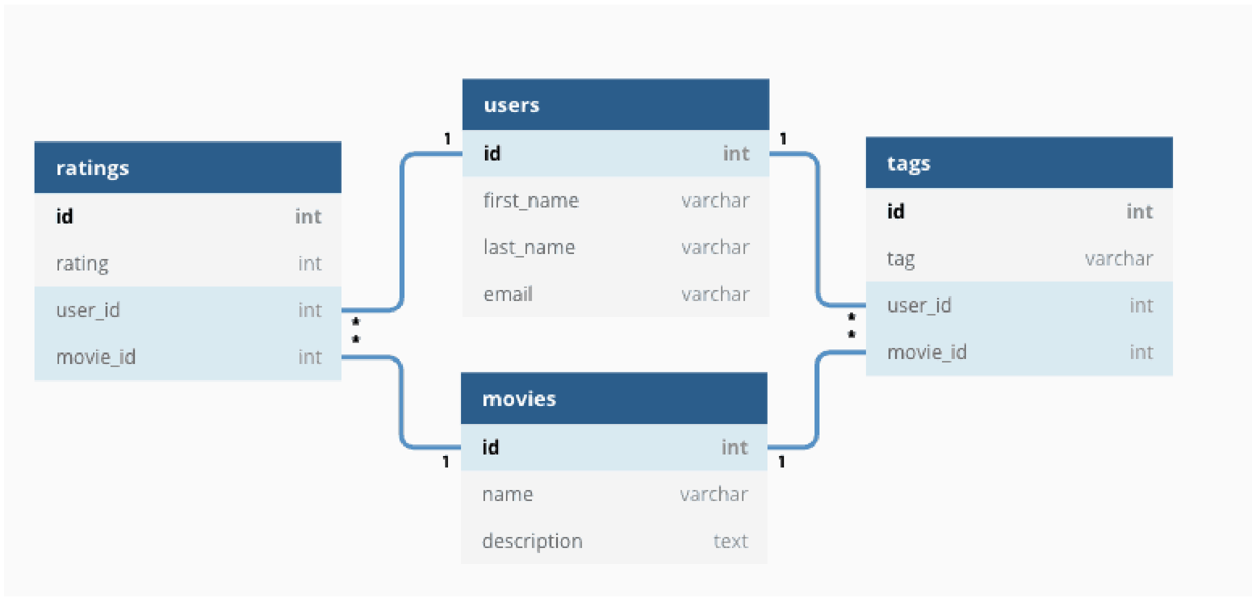
Nature of relational database

- A relational database is a type of database that stores and provides access to data points that are related to one another.
- A relational database organizes data into tables which can be linked—or related—based on data common to each.
- This capability enables you to retrieve an entirely new table from data in one or more tables with a single query. It also allows you and your business to better understand the relationships among all available data and gain new insights for making better decisions or identifying new opportunities.
- Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables.
- In a relational database, each row in the table is a record with a unique ID called the key.
- The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.
- Here's a simple example of two tables a small business might use to process orders for its products. The first table is a customer info table, so each record includes a customer's name, address, shipping and billing information, phone number, and other contact information. Each bit of information (each attribute) is in its own column, and the database assigns a unique ID (a key) to each row. In the second table—a customer order table—each record includes the ID of the customer that placed the order, the product ordered, the quantity, the selected size and color, and so on—but not the customer's name or contact information.
- These two tables have only one thing in common: the ID column (the key).
- But because of that common column, the relational database can create a relationship between the two tables.
- Then, when the company's order processing application submits an order to the database, the database can go to the customer order table, take the correct information about the product order, and use the customer ID from that table to look up the customer's billing and shipping information in the customer info table.
- The warehouse can then take the correct product, the customer can receive timely delivery of the order, and the company can get paid.



Advantages of Relational Database

1. Speed

Even though a relational database is poor in terms of performance, still its speed is considerably higher because of its ease and simplicity. And also various optimizations that are included in a relational database further increases its speed. So all the applications will run with appropriate speed when used in a relational database.

2. Security

Since there are several tables in a relational database, certain tables can be made to be confidential. These tables are protected with username and password such that only authorized users will be able to access them. The users are only allowed to work on that specific table.

3. Simplicity

Compared to other types of network models, a relational database model is much simpler. It is free from query processing and complex structuring. As a result, it does not require any complex queries. A simple SQL query is sufficient enough for handling.

4. Accessibility

Unlike other types of databases, a relational database does not require any specific path for accessing the data. Even modifying data in the relevant column is made easy. So whatever the outcome shown is appropriate to the user.

5. Accuracy

As mentioned earlier, relational databases use primary keys and foreign keys to make the tables interrelated to each other. Thus, all the data which is stored is non-repetitive. Which means that the data does not duplicate. Therefore, the data stored can be guaranteed to be accurate.

6. Multi User

Multiple users will be able to access a relational database at the same time. Even if the data is updated, the users can access them conveniently. Hence, the crashes happening from multi access are possibly prevented.

Disadvantages of Relational Database

1. Cost

The underlying cost involved in a relational database is quite expensive. For setting up a relational database, there must be separate software which needs to be purchased. And a professional technician should be hired to maintain the system. All these can be costly, especially for businesses with a small budget.

2. Performance

The performance of the relational database depends on the number of tables. If there are more tables, the response given to the queries will be slower. Additionally, more data presence not only slows down the machine, it eventually makes it complex to find information. Thus, a relational database is known to be a slower database.

3. Physical Storage

A relational database also requires a tremendous amount of physical memory since it is with rows and columns. Each of the operations depend on separate physical storage. Only through proper optimization, the targeted applications can be made to have maximum physical memory.

4. Complexity

Although a relational database is free from complex structuring, occasionally it may become complex too. When the amount of data in a relational database increases, it eventually makes the system more complicated. Each and every data is complex since the data is arranged using common characteristics.

5. Information Loss

Large organizations tend to use more database systems with more tables. This information can be used to be transferred from one system to another. This could pose a risk of data loss.

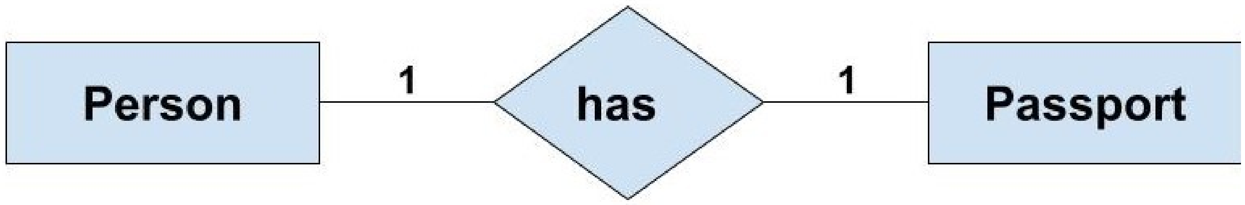
6. Structure Limitations

The fields that are present on a relational database are with limitations. Limitations in essence means that it cannot accommodate more information. However, if more information is provided, it may lead to data loss. Therefore, it is necessary to describe the exact amount of data volume which the field will be given.

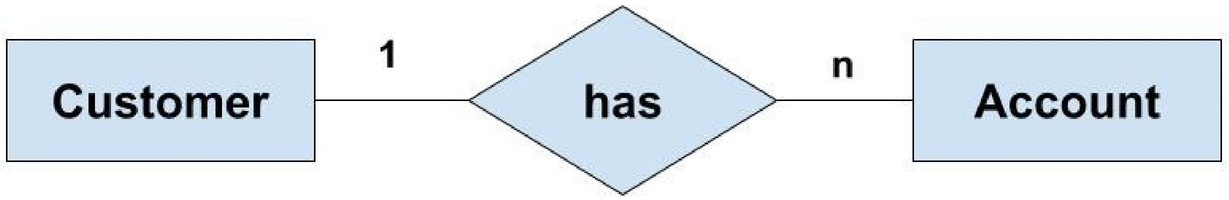
Types of relationships in RDBMS

- Any association between two entity types is called a relationship. Entities take part in the relationship. It is represented by a diamond shape.
- For example, A teacher teaches students. Here, "teaches" is a relationship and this is the relationship between a Teacher entity and a Student entity.
- There are three types of relationships that can exist between two entities.
    - One-to-One Relationship
    - One-to-Many or Many-to-One Relationship
    - Many-to-Many Relationship
- **One-to-One Relationship :**
    - Such a relationship exists when each record of one table is related to only one record of the other table.

    - For example, If there are two entities ‘Person’ (Id, Name, Age, Address)and ‘Passport’(Passport_id, Passport_no). So, each person can have only one passport and each passport belongs to only one person.
    - Such a relationship is not very common. However, such a relationship is used for security purposes. In the above example, we can easily store the passport id in the ‘Person’ table only. But, we make another table for the ‘Passport’ because the Passport number may be sensitive data and it should be hidden from certain users. So, by making a separate table we provide extra security that only certain database users can see.



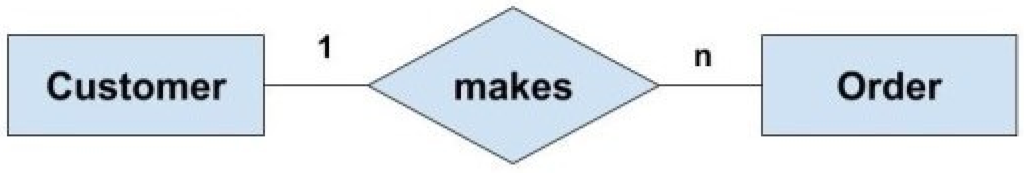
- **One-to-Many or Many-to-One Relationship**
    - Such a relationship exists when each record of one table can be related to one or more than one record of the other table. This relationship is the most common relationship found. A one-to-many relationship can also be said as a many-to-one relationship depending upon the way we view it.
    - For example, If there are two entity types ‘Customer’ and ‘Account’ then each ‘Customer’ can have more than one ‘Account’ but each ‘Account’ is held by only one ‘Customer’. In this example, we can say that each Customer is associated with many accounts. So, it is a one-to-many relationship. But, if we see it the other way i.e many Account is associated with one Customer then we can say that it is a many-to-one relationship.

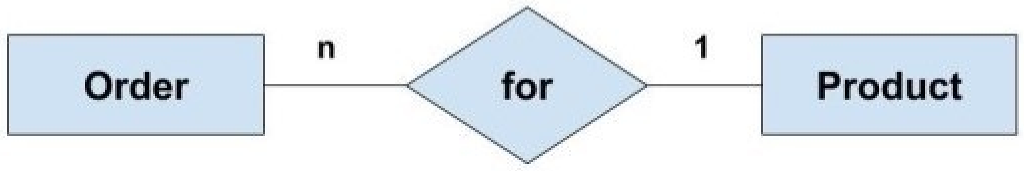


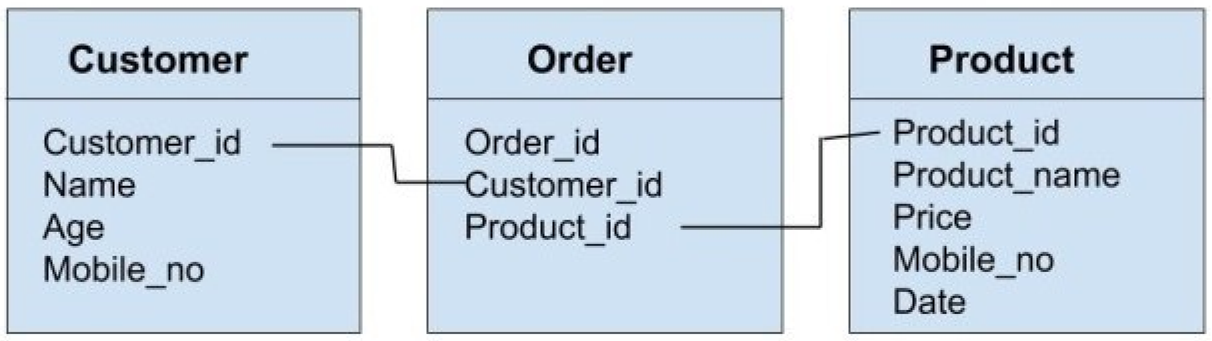
- **Many-to-Many Relationship**
    - Such a relationship exists when each record of the first table can be related to one or more than one record of the second table and a single record of the second table can be related to one or more than one record of the first table. A many-to-many relationship can be seen as a two one-to-many relationship which is linked by a 'linking table' or 'associate table'. The linking table links two tables by having

fields which are the primary key of the other two tables. We can understand this with the following example.

- Example: If there are two entity types ‘Customer’ and ‘Product’ then each customer can buy more than one product and a product can be bought by many different customers.
- Now, to understand the concept of the linking table here, we can have the ‘Order’ entity as a linking table which links the ‘Customer’ and ‘Product’ entity. We can break this many-to-many relationship in two one-to-many relationships. First, each ‘Customer’ can have many ‘Order’ whereas each ‘Order’ is related to only one ‘Customer’. Second, each ‘Order’ is related to only one Product where there can be many orders for the same Product.
- The above concept of linking can be understood with the help of taking into consideration all the attributes of the entities 'Customer', 'Order' and 'Product'. We can see that the primary keys of both 'Customer' and 'Product' entities are included in the linking table i.e 'Order' table. These keys act as foreign keys while referring to the respective table from the 'Order' table.







Database operations CRUD

CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: create, read, update and delete. Persistent storage refers to any data storage device that retains power after the device is powered off, such as a hard disk or a solid-state drive. In contrast, random access memory and internal caching are two examples of volatile memory - they contain data that will be erased when they lose power.

A relational database consists of tables with rows and columns. In a relational database, each row of a table is known as a tuple or a record. Each column of the table represents a specific attribute or field. The four CRUD functions can be called by users to perform different types of operations on selected data within the database. This could be accomplished using code or through a graphical user interface. Let's review each of the four components in-depth to fully appreciate their collective importance of facilitating database interactions.

1. **Create**

The create function allows users to create a new record in the database. In the SQL relational database application, the Create function is called INSERT. A user can create a new row and populate it with data that corresponds to each attribute, but only an administrator might be able to add new attributes to the table itself.

2. **Read**

The read function is similar to a search function. It allows users to search and retrieve specific records in the table and read their values. Users may be able to find desired records using keywords, or by filtering the data based on customized criteria. For example, a database of cars might enable users to type in "1996 Toyota Corolla", or it might provide options to filter search results by make, model and year.

3. **Update**

The update function is used to modify existing records that exist in the database. To fully change a record, users may have to modify information in multiple fields. For example, a restaurant that stores recipes for menu items in a database might have a table whose attributes are "dish", "cooking time", "cost" and "price". One day, the chef decides to replace an ingredient in the dish with something different. As a result, the existing record in the database must be changed and all of the attribute values changed to reflect the characteristics of the new dish.

4. **Delete**

The delete function allows users to remove records from a database that is no longer needed. Most databases have a delete function that allows users to delete one or more records from the database. Some relational database applications may permit users to perform either a hard delete or a soft delete. A hard delete permanently removes records from the database, while a soft delete might simply update the status of a row to indicate that it has been deleted while leaving the data present and intact.

SQL

SQL is a standard language for accessing and manipulating databases.

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Types of SQL Statements

Here are five types of widely used SQL queries.

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transaction Control Language (TCL)
- Data Query Language (DQL)

ACID Properties

- A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.
  - **A**tomicity − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
  - **C**onsistency − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
  - **I**solation − In a database system where more than one transaction is being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and

executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

- **D**urability − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

Some of The Most Important SQL Commands

SELECT - extracts data from a database

UPDATE - updates data in a database

DELETE - deletes data from a database

INSERT INTO - inserts new data into a database

CREATE DATABASE - creates a new database

ALTER DATABASE - modifies a database

CREATE TABLE - creates a new table

ALTER TABLE - modifies a table

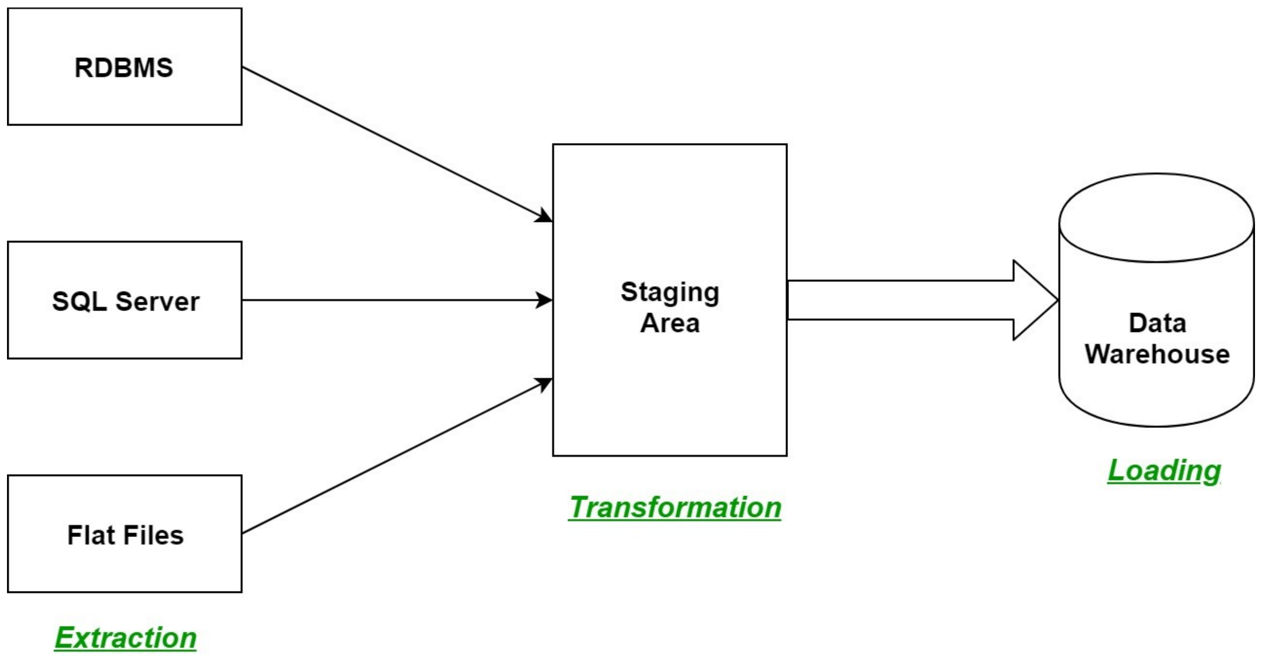
DROP TABLE - deletes a table

CREATE INDEX - creates an index (search key)

DROP INDEX - deletes an index

Identify the meaning of ETL

ETL is a process in Data Warehousing and it stands for **Extract**, **Transform** and **Load**. It is a process in which an ETL tool extracts the data from various data source systems, transforms it in the staging area, and then finally, loads it into the Data Warehouse system.



Let us understand each step of the ETL process in-depth:

1. **Extraction:**
   The first step of the ETL process is extraction. In this step, data from various source systems is extracted which can be in various formats like relational databases, No SQL, XML, and flat files into the staging area. It is important to extract the data from various source systems and store it into the staging area first and not directly into the data warehouse because the extracted data is in various formats and can be corrupted also. Hence loading it directly into the data warehouse may damage it and rollback will be much more difficult. Therefore, this is one of the most important steps of the ETL process.
2. **Transformation:**
   The second step of the ETL process is transformation. In this step, a set of rules or functions are applied on the extracted data to convert it into a single standard format. It may involve following processes/tasks:
   - Filtering – loading only certain attributes into the data warehouse.
   - Cleaning – filling up the NULL values with some default values, mapping U.S.A, United States, and America into USA, etc.
   - Joining – joining multiple attributes into one.
   - Splitting – splitting a single attribute into multiple attributes.
   - Sorting – sorting tuples on the basis of some attribute (generally key-attribute).
3. **Loading:**
   The third and final step of the ETL process is loading. In this step, the transformed data is finally loaded into the data warehouse. Sometimes the data is updated by loading into the data warehouse very frequently and sometimes it is done after longer but regular intervals. The rate and period of loading solely depends on the requirements and varies from system to system.

NoSQL Databases

Databases can be divided in 3 types:

- RDBMS (Relational Database Management System)
- OLAP (Online Analytical Processing)
- NoSQL (recently developed database)

What is NoSQL Database

- NoSQL Database is used to refer to a non-SQL or non relational database.
- Relational databases also had a problem that they could not handle big data. Due to this problem there was a need for a database which could handle every type of problem. Then NoSQL was developed.
- NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would be "NoREL", NoSQL caught on.
- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.
- NoSQL Database is a non-relational Data Management System that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.
- It provides a mechanism for storage and retrieval of data other than the tabular relations model used in relational databases. NoSQL databases don't use tables for storing data. It is generally used to store big data and real-time web applications.

Advantages of NoSQL

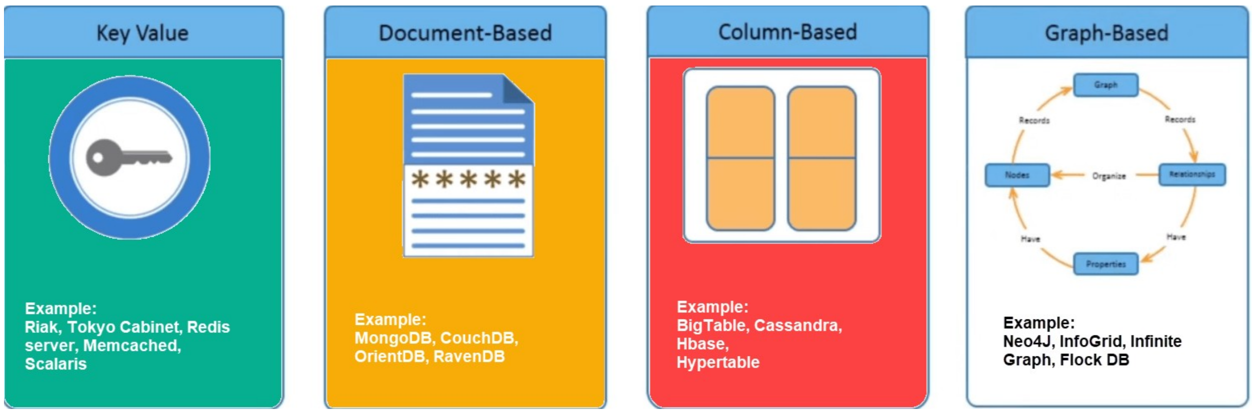
- It supports query language.
- It provides fast performance.
- It provides horizontal scalability.

Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types:

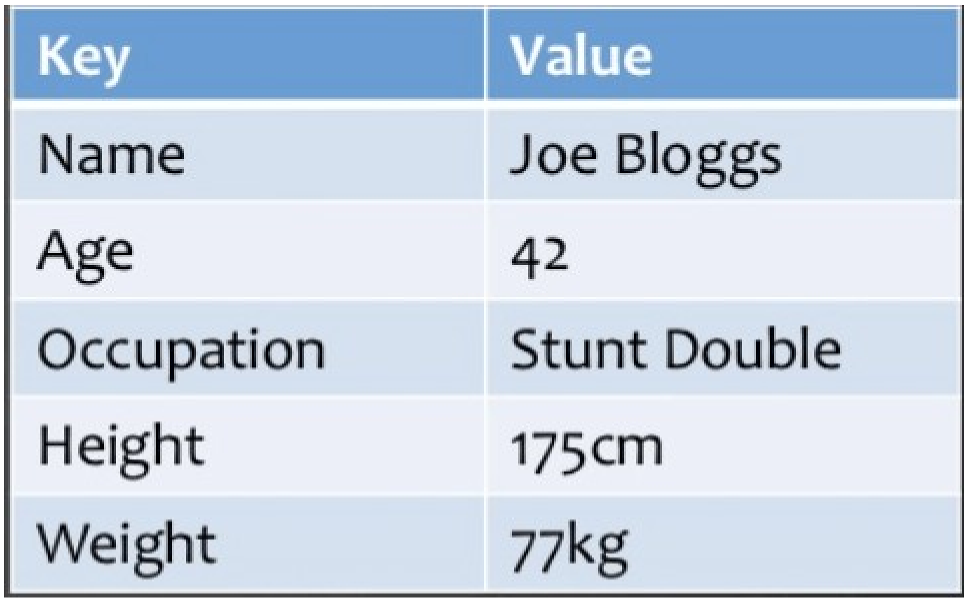
- Key-value pair
- Column-oriented
- Graph-based
- Document-oriented.

Every category has its unique attributes and limitations. None of the above-specified databases is better to solve all the problems. Users should select the database based on their product needs.


Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Records
Nodes
Organize
Relationships
Have
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key Value Pair Based

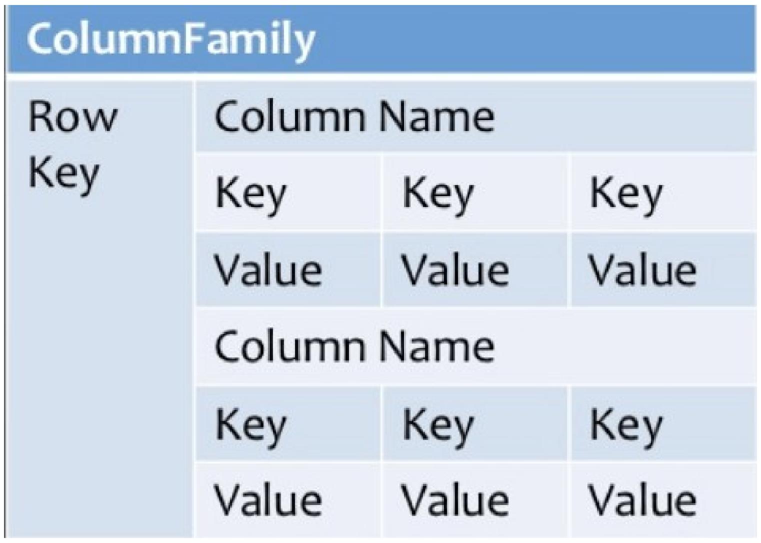
- Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a string, binary etc.
- For example, a key-value pair may contain a key like “Website” associated with a value like “Guru99”.
- This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.



Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Column-based

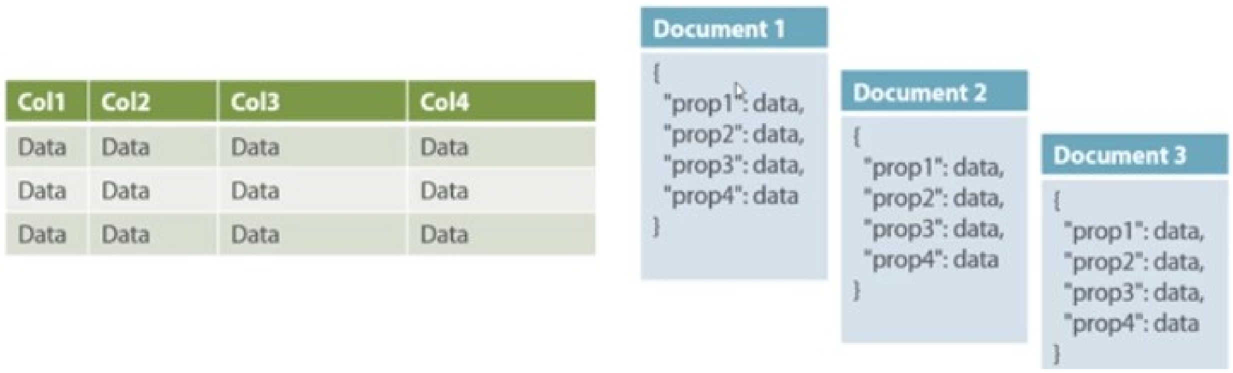
- Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.
- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,



Column based NoSQL database

Document-Oriented:

- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document.
- In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON.
- Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have a data store like a JSON object. You do not require a definition which makes it flexible.
- The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications.
- It should not be used for complex transactions which require multiple operations or queries against varying aggregate structures.



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

```
Document 1
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

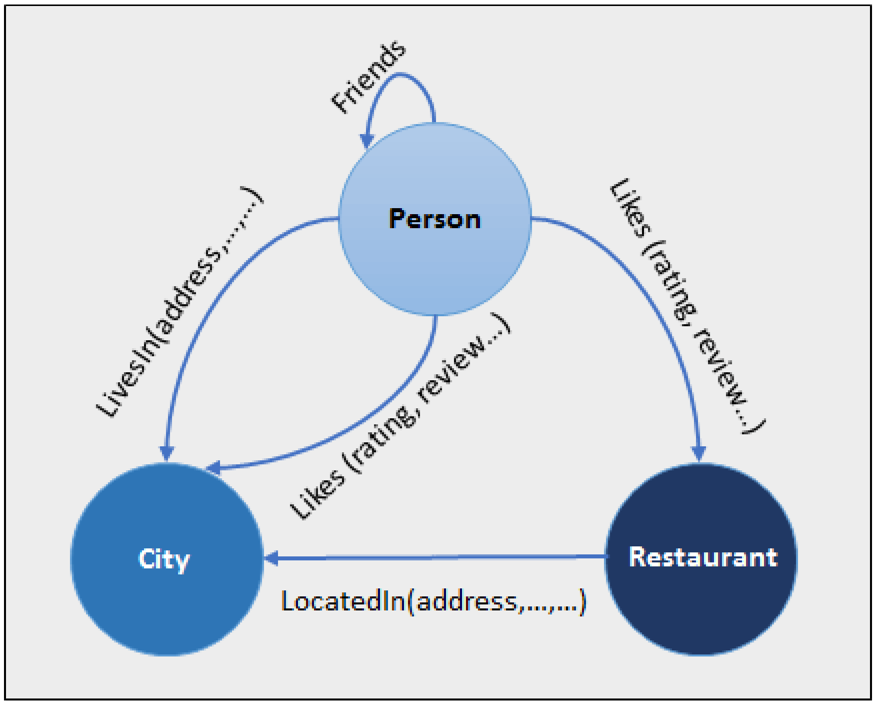
```
Document 2
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

```
Document 3
{
 "prop1": data,
 "prop2": data,
 "prop3": data,
 "prop4": data
}
```

Relational Vs. Document

Graph-Based

- A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.
- Compared to a relational database where tables are loosely connected, a Graph database is multi-relational in nature.
- Traversing relationships is fast as they are already captured into the DB, and there is no need to calculate them.
- Graph based database mostly used for social networks, logistics, spatial data.


Friends
Person
LivesIn(address,.....,...)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

What MongoDB is used for?

MongoDB is an open source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.