Database

File-Based Storage System/File Based Database Management System:
A file-based storage system is a database management system where data is stored in the form of files. This system allows access to a single file at a time. It is a collection of flat files(single tables stored in one file) that do not carry any correlation with other files in the system.
Example: Microsoft NTFS(New Technology File System), Hierarchical File System of Apple.

Challenges of File-Based Storage System
This storage system has multiple disadvantages like:

1. Data Redundancy:
   In a traditional file management system, redundancy occurs because the same data is stored at different places. If the data is updated or replaced at someplace and not updated at other places, the same data would have multiple copies with differences and it would be difficult to find out the latest data. This causes a threat to data security and integrity. This can cause multiple data anomalies like delete anomaly, insert anomaly and read anomaly. This also causes data inconsistency.
2. Poor data security:
   Because of data redundancy, multiple copies of the same data are present in the system which causes a possibility of data breaches by unauthorised users. Therefore, it is a serious threat to data security.
3. Slow:
   The traditional file management system is not very quick because it needs a lot of ad-hoc queries and more extensive programming comparatively for report generation. Therefore, only programmers can deal with this storage system easily.
4. Not so efficient data retrieval:
   While incorporating this file-based management system in today's applications, there would be multiple performance issues and the speed is very slow. Due to these challenges, data retrieval is not very efficient.

To solve these problems, a new database management system was introduced known as RDBMS(Relational Database Management System) to solve these problems.

Relational Database Management System(RDBMS):
It is one of the most popular database management systems. RDBMS is a software that performs all data-related operations on a relational database like a store, manage, query, and retrieve data. It is based on the relational model. The significant components of RDBMS are tables. Data is represented in the form of tables. The relationship between the two tables is represented by foreign keys.
<u>Example</u>: MYSQL, Oracle etc.

Advantages of Relational Database Management System over File System

1. <u>Data redundancy and inconsistency</u>
   Redundancy refers to the concept of data repetition, which means that any data item may have multiple copies. Because each user sets and maintains the files required for a specific application to execute, the file system is unable to control data redundancy. It's possible that two users are sharing the same files and data for different programmes. As a result, changes performed by one user do not appear in files utilised by other users, resulting in data inconsistency. RDBMS, on the other hand, manages redundancy by keeping a single data repository that is defined once and accessed by multiple users. Data remains constant because there is no or little redundancy.

2. <u>Data sharing</u>
   The file system does not allow data sharing or it is too complicated. Due to the centralised structure in RDBMS, data may be simply exchanged.

3. <u>Data concurrency</u>
   When more than one user accesses the same data at the same time, this is referred to as data concurrency. Anomalies occur when one user's edits are overwritten by changes made by another user. There is no method in the file system to prevent abnormalities. A locking system is provided by RDBMS to prevent abnormalities from occurring.

4. <u>Data searching</u>

Each file system search activity necessitates the creation of a separate application programme. RDBMS, on the other hand, has built-in searching capabilities. To access data from the database, the user merely needs to submit a short query.

5. Data integrity
   Before putting data into a database, some constraints may need to be applied to the data. There is no process in the file system to check these constraints automatically. RDBMS, on the other hand, ensures data integrity by enforcing user-defined restrictions on data.

Challenges of Relational Database Management System

This storage system had multiple disadvantages like:

1. Rigid Schema:
   In case the data schema is dynamic and changes, RDBMS would not be compatible.
2. Limited Scaling Patterns:
   The limitation here is that scaling patterns are very limited, unlike NoSQL DBs.
3. Cost:
   RDBMS is comparatively expensive software.
4. Skills:
   The data administrator must be skilled in order to use this storage system.

We'll go through the two basic types of databases:

1. Relational Databases (SQL based).
2. NoSQL databases

SQL Database

SQL is a computer language that was created for managing data stored in a relational database management system. The data in relational DBMSs appear as tables of rows and columns with a well-defined structure and dependencies.

SQL databases require little engineering effort to secure because of their integrated structure and data storage method. They're a fantastic fit for creating and maintaining complicated software solutions where every interaction has a variety of outcomes. ACID compliance is one of the SQL foundations (Atomicity, Consistency, Isolation, Durability). If we are creating eCommerce or financial apps, for example, where database integrity is crucial, ACID compliance is the way to go.

Some examples of SQL databases

- MySQL
- Oracle
- PostgreSQL

<u>Some Applications of SQL are</u>

- Developers and DBAs (Database Administrators) use SQL to create Data Integration Scripts.
- It is used to handle analytical queries in order to evaluate data and derive insights from it.
- Information Retreival
- Insertion, Deletion, and Updation are examples of data and database table manipulation.

Advantages for using SQL Database:

1. It has a straightforward structure that corresponds to the majority of data types seen in most programmes.
2. It makes use of SQL, which is widely used and enables JOIN operations by default.
3. Allows for quick data updates. Because the entire database is saved on a single machine and relationships between entries are used as pointers, we can update a record once and have all of its associated records updates at the same time.
4. Atomic transactions are also supported by relational databases.

NoSQL Database

It stands for "non-SQL" database, or we can say that it is a non-relational database. It is complementary to RDBMS. It is built on the principle of a distributed system. Therefore, they are natively scalable. NoSQL is the umbrella term comprising of four different types of databases.
<u>Example:</u> MongoDB, Redis etc.

Advantages of NoSQL Database:

1. Semi-structured data

2. Dynamic or flexible schema
3. Non-relational data
4. No need for complex joins
5. Store many TB (or PB) of data
6. Very data-intensive workload
7. Very high throughput for IOPS

Example 1:

You wish to develop an e-commerce website that would contain multiple databases and tables storing user's login information and staff details including ID, date of joining, phone number, address, email id.

What would be the suitable database choice here out of SQL or NoSQL databases?

Answer 1: The appropriate database here is an SQL database since the data is structured primarily. Transaction-oriented systems, such as customer relationship management tools, accounting software, and e-commerce platforms, benefit greatly from SQL databases.

Example 2:

Let's pretend you're working on the next Google Analytics. You've decided to monitor IP addresses, browsers, and device types. However, you later realise that you might want to keep track of Browser Size as well. It's not straightforward to add an extra column to your table because analytics databases can include millions or billions of records. It would simply be too time-consuming.

What will you choose from SQL and NoSQL databases for this situation?

Answer 2: We should select the NoSQL database here since it has a dynamic and flexible schema that would allow us to add and append rows in the database without changing and re-ordering each row of the tracking table.

The four different types of NoSQL Databases are:

1. Key-value store
2. Document store
3. Column-oriented database
4. Graph database

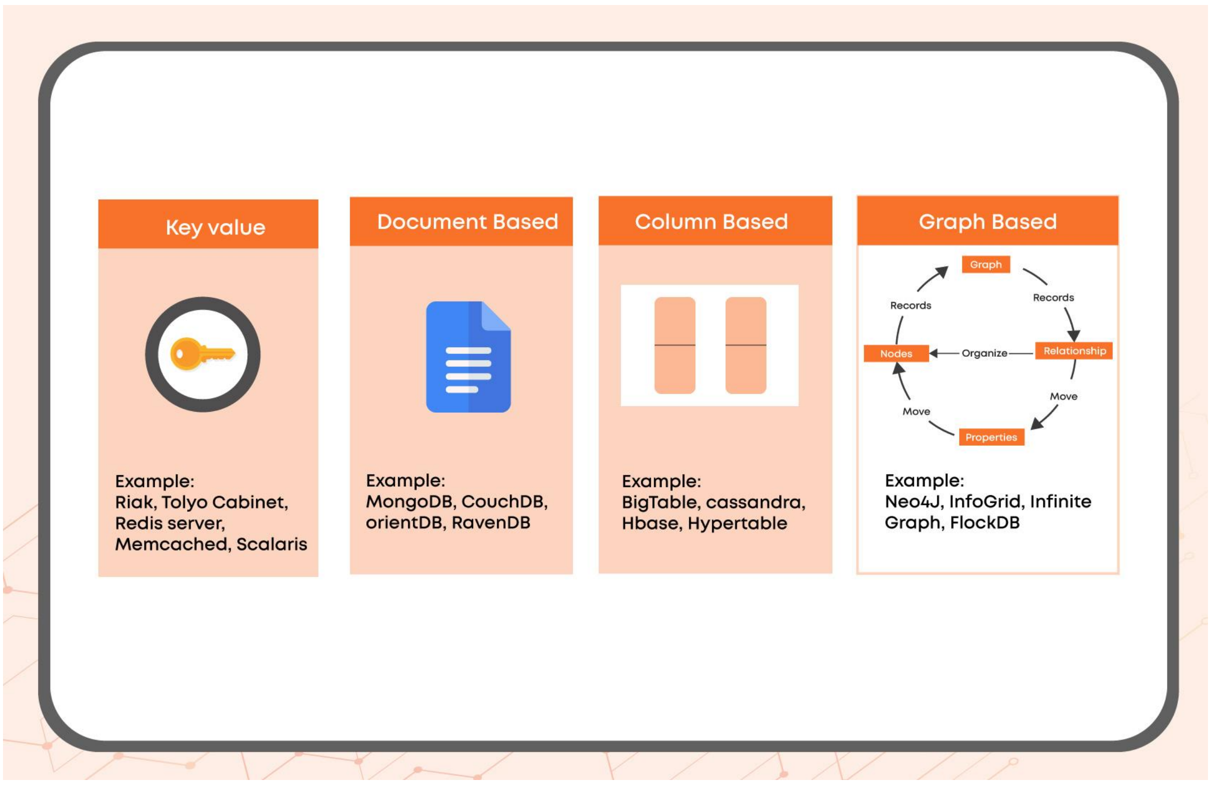
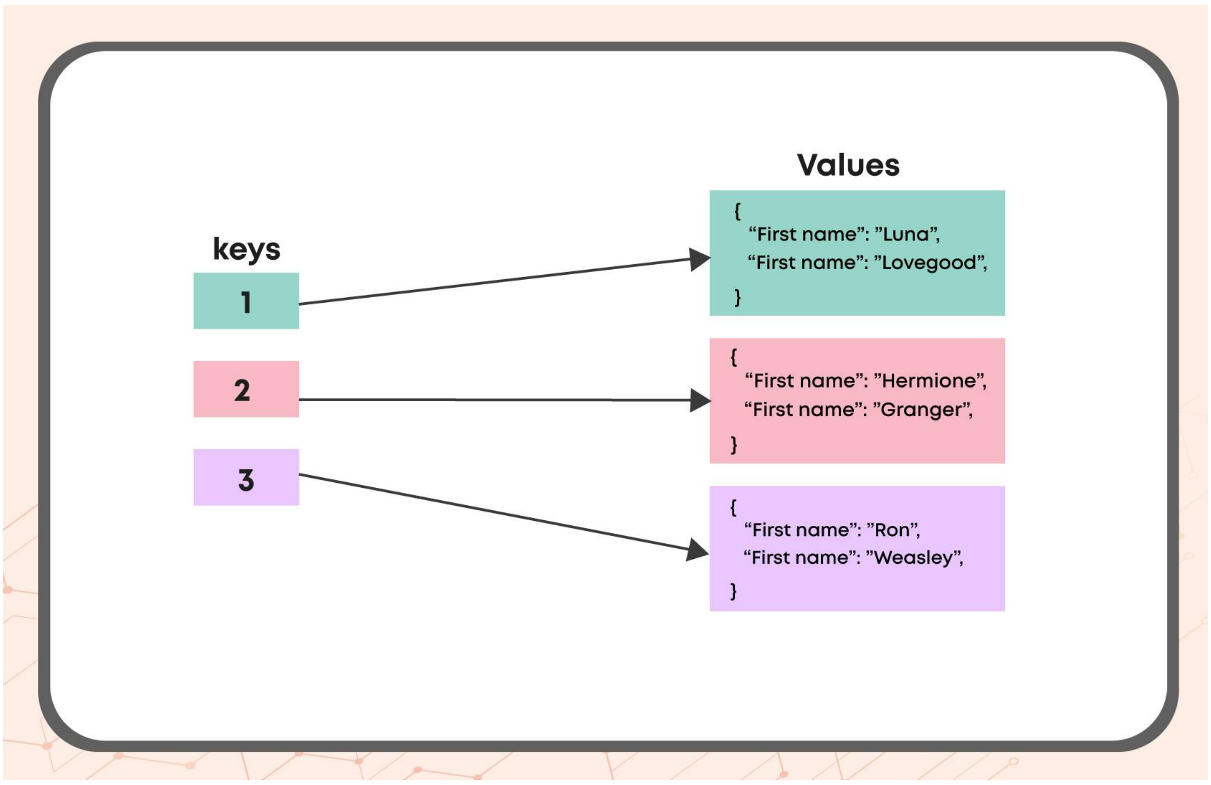


Fig: Different types of NoSQL Databases

1. **<u>Key-Value Database:</u>**

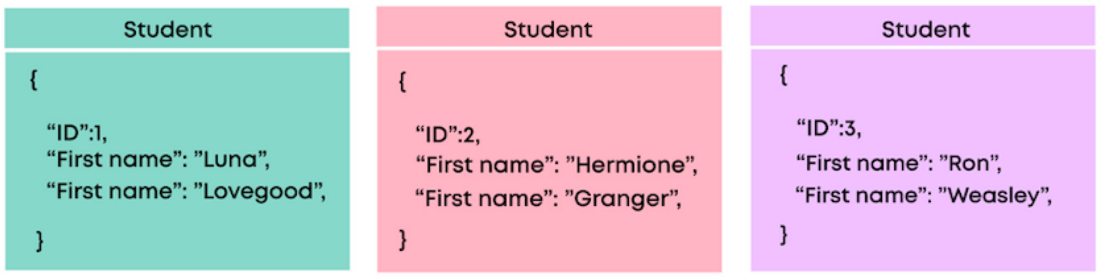
   It stores data in pairs of keys and values. Therefore, it is a non-relational database because the data is not stored in a table but in the form of the key-value method. It is widely used as a caching solution. Suppose we need to store the names of mentors working at Coding Ninjas. The “name” would be the key, and the value would be different names of mentors.
   The TTL column (time to live) is a helpful feature in this database; it may be set differently for each entry and indicates when it should be destroyed from the database.
   <u>Example:</u> Redis

   <u>When should we use Key-Value DB?</u>
   Because it is fast and does not require extensive queries, it is primarily used for caching. The TTL function is also highly beneficial for caching.
   It can also be used for any other type of data with a key-value format that requires quick querying.


Values
keys
1
2
3
{ "First name": "Luna", "First name": "Lovegood", }
{ "First name": "Hermione", "First name": "Granger", }
{ "First name": "Ron", "First name": "Weasley", }

2. **Document Database:**
   Data is stored in the form of JSON like documents. It combines the concepts of RDBMS and NoSQL databases. It combines the relationship concept from RDBMS and dynamic schema and horizontal scaling from NoSQL databases.
   Example: MongoDB

   When should we use Document DB?
   Data analysis: This database facilitates parallel computations because separate records are not logically or structurally dependent on one another.
   This makes it simple to run big data analytics on our data.

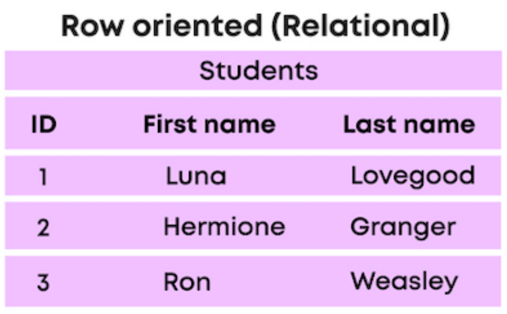

Student
{ "ID":1, "First name": "Luna", "First name": "Lovegood", }
Student
{ "ID":2, "First name": "Hermione", "First name": "Granger", }
Student
{ "ID":3, "First name": "Ron", "First name": "Weasley", }

3. **Columnar Database:**

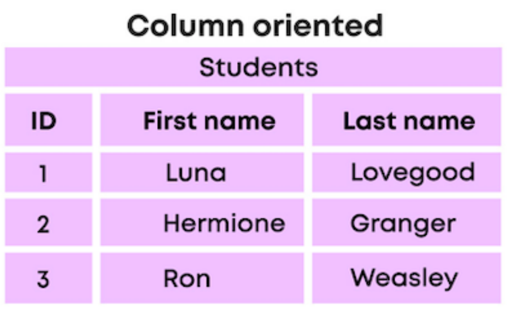
A columnar database is one in which the columns are stored together instead of rows. Most of the operations are column-oriented in an application, and because of that, the aggregation in such databases is rapid. It is widely used for running analytical queries or Time Series Data.
Example: Cassandra

When should we use Columnar DB?
When we query on a subset of your data's columns.
Because it just needs to read these specific columns, columnar DB conducts such queries quickly (while row-based DB would have to read the entire data).

- Each column represents a feature, which is frequent in data science. Data scientist frequently trains models with subsets of the features and frequently examine feature-score relationships (correlation, variance, significance).
- They often save a lot more attributes in our logs database but only use a few in each query, which is also common with logs.

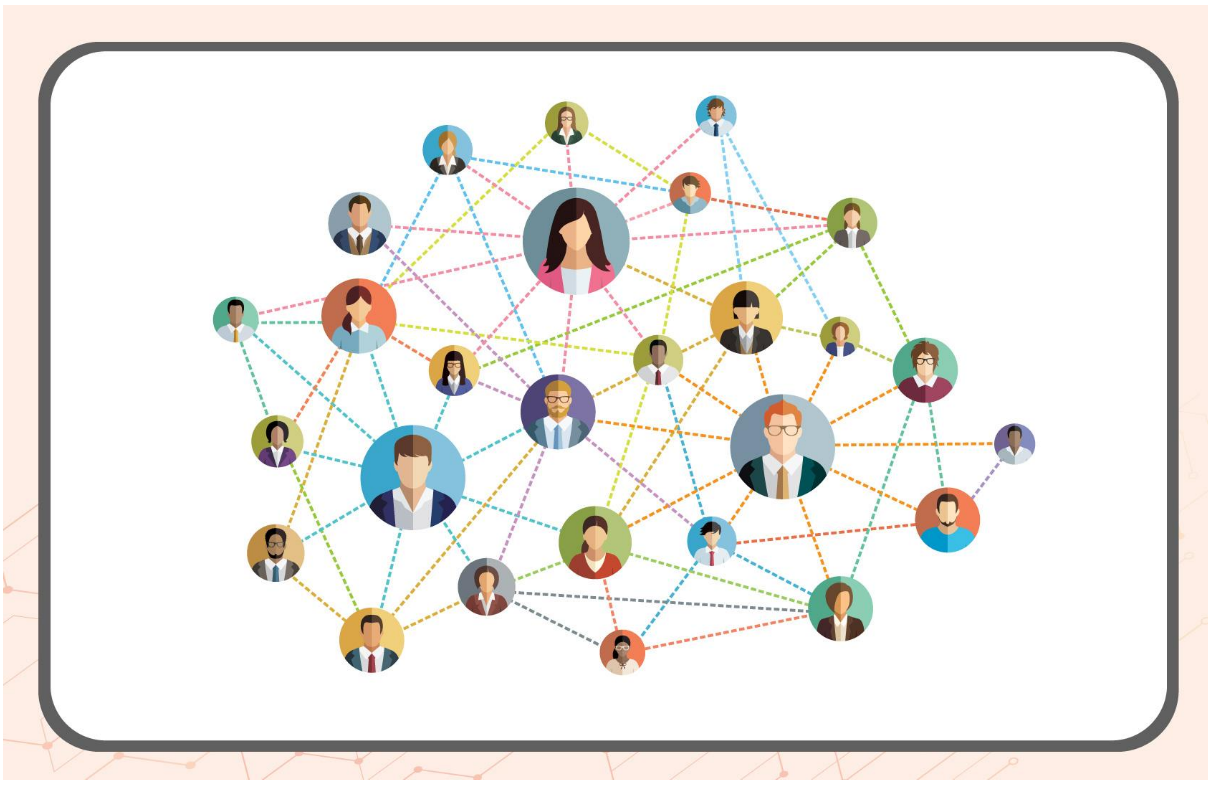

Row oriented (Relational)

Students		
ID	First name	Last name
1	Luna	Lovegood
2	Hermione	Granger
3	Ron	Weasley


Column oriented

Students		
ID	First name	Last name
1	Luna	Lovegood
2	Hermione	Granger
3	Ron	Weasley

4. **Graph Database:**
Graph database represents and stores entities and relationships in the form of graph data structure. It is majorly used for social networks and applications incorporating relationships with entities like google maps
.



Example: Neo4j

When should we use Graph DB?
When your data is in the form of a graph, such as knowledge graphs or social networks.

Summary on usage

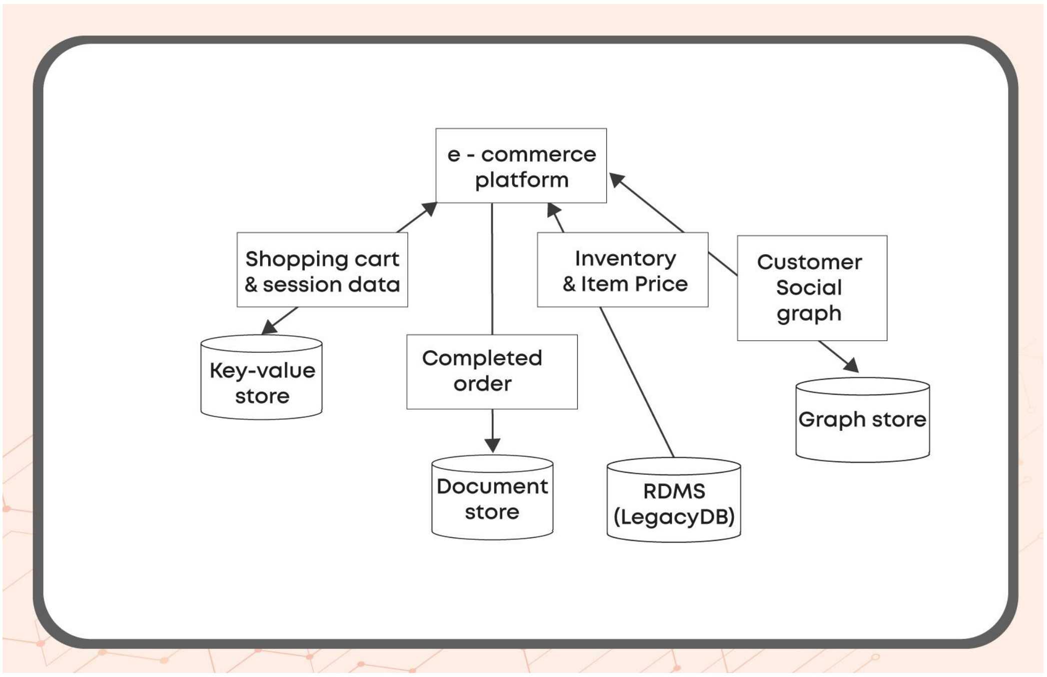
Key-Value Database	Caching
Graph Database	A graph like data for example social networks.
Columnar Database	If querying on columns is required
Document Database	Flexible Schema or we might need to change the schema in future

Polyglot Persistence

Polyglot persistence refers to the use of different data storage technologies across an application or within smaller components of an application to meet varying data storage demands.
Polyglot persistence occurs when one particular application or project uses multiple databases depending on varied data requirements. Different databases can be used for various components of the single application to meet the individual requirements best. Because of varying needs and increasing competition, most of the applications use polyglot persistence these days.

Example: Let's take the example of an e-commerce website like Flipkart. It has multiple functions like inventory management, shopping cart management, payments etc. For better performance, it uses different databases instead of storing all the information in a single database.
It uses different databases like:

1. Oracle for payments: We will need ACID(Atomicity, Consistency, Isolation, Durability) properties here provided by SQL database. It will provide lightning-fast transactions, easy horizontal scaling, and support for JSON documents within records as well as JSON queries.
2. MongoDB for storing data of the product: Flexibility of MongoDB documents can be used here. Each MongoDB document can store data in the form of sophisticated JSON constructs. MongoDB is therefore perfect for storing almost anything, including very big catalogues with thousands of versions per item.
3. Redis(key-value database) for shopping e-cart content: The key-value database can be used for storing cart content. If we wanted to store user session data, shopping cart information, and user preferences, we could just store all of them in the same bucket with a single key and single value for all of these objects.
4. ES(Document DB) for text-based search functions: Structured search is available in NoSQL databases that use document stores, and it preserves the best capabilities of Boolean and full-text keyword search.
5. Cassandra (Document Database) for the data warehouse: A columnar database stores data by columns rather than by rows, which makes it suitable for analytical query processing, and thus for data warehouses.



Normalisation:
Normalisation is the process of organising the data in different tables. Data is stored in multiple tables to avoid redundancy and data anomalies.
We need to perform Normalization on our database to reduce Data Redundancy. It minimizes redundancy using certain rules or sets of rules.
There are many types of normal forms, although we are going to focus on 1Nf, 2NF, 3NF and BCNF (also known as 3.5 NF).

The most commonly used normal forms are:

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)
4. Boyce & Codd normal form (BCNF)

Types of Normal Forms:

1. **First Normal Form (1NF):** This is Step 1 of the Normalisation Process.
   For a Relation/table to justify 1NF it needs to satisfy 4 basic conditions:
   - Each attribute should contain atomic values. (i.e. No multivalued attributes)
   - Each Value stored in an attribute should be of the same type.
   - All the attributes in a table should have unique names.

- The order of the data stored in the table doesn't matter.

2. **Second Normal Form:** For a Relation/table to justify 2NF it needs to satisfy 2 rules:
   - It should be in First Normal Form.
   - It should not have any partial dependencies i.e. when a nonprime attribute is derivable from only a part of a candidate key.

3. **Second Normal Form:** For a Relation/table to justify 2NF it needs to satisfy 2 rules:
   - It should be in First Normal Form.
   - It should not have any partial dependencies i.e. when a nonprime attribute is derivable from only a part of a candidate key.

4. **Boyce-Codd Normal Form (BCNF):** It is an extension of 3NF and is also known as the 3.5 Normal Form.
   For a table to be in the Boyce-Codd Normal Form, it should satisfy 2 rules:
   - It should be in the Third Normal Form.
   - A prime attribute shouldn't be dependent on a non-prime attribute. (i.e. if M→ N, then M is a superkey)

Denormalization:
It is the opposite of Normalisation. Normalisation is breaking the data and organising it in different tables whereas denormalisation is combining the data. Denormalisation combines the data and organises it in a single table. Denormalization is the process of adding redundant data in the normalised relational database to optimise the performance.

Benefits of denormalisation:
1. Faster data read operations
2. Simpler and more accessible to query
3. High data availability
4. Requires less computation
5. Reduced network calls
6. High data accessibility

Challenges of decentralisation:

1. Slow write operations
2. Increases complexity
3. Wastage of memory
4. Data inconsistency

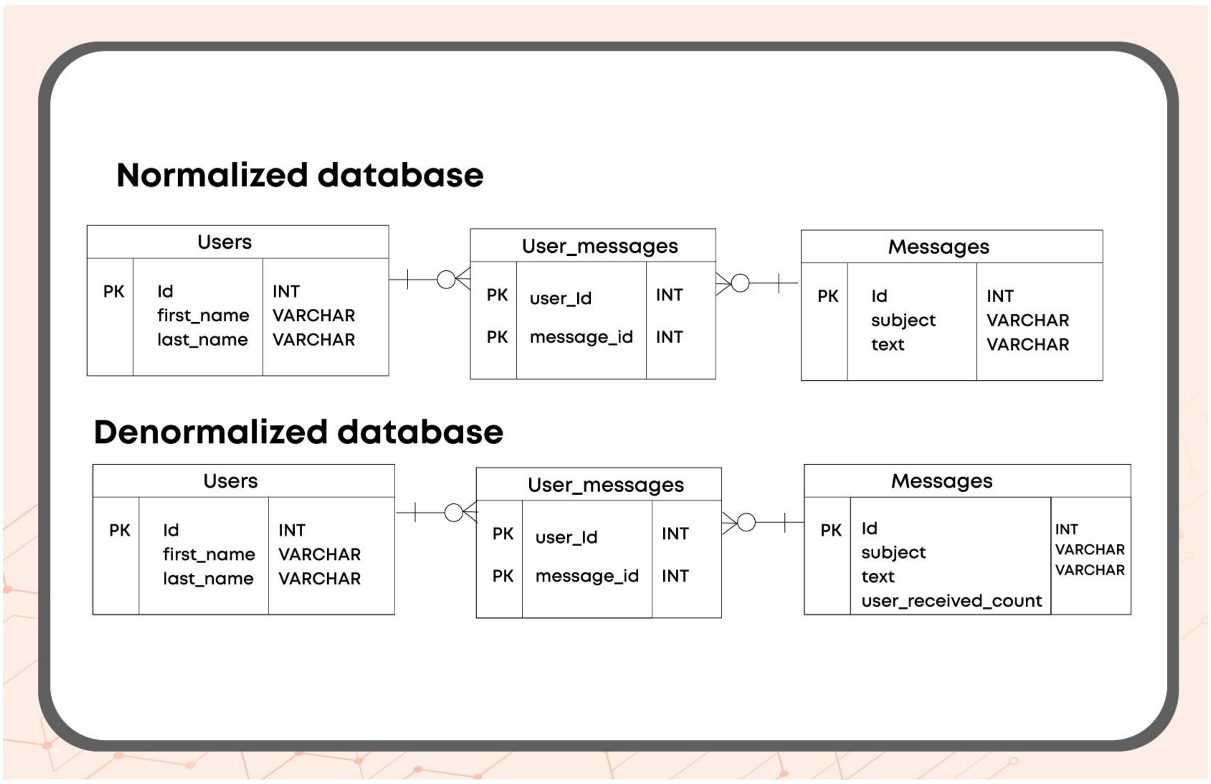


Fig: Normalization vs Denormalization