Databases and Types of Databases

SQL Databases

- SQL (Structured Query Language) database is a type of database that follows the relational model, where data is organised into tables, and each table contains rows and columns.
- SQL databases are used for applications that require complex queries and transactional consistency.
- To maintain data integrity it follows ACID properties.

  ACID Property -

  - **A**tomicity -
    - Ensures that a transaction is treated as a single unit, meaning that either all operations within the transaction are completed successfully, or none are.
    - If any part of the transaction fails, the entire transaction is rolled back to its previous state, preventing partial updates that could leave the data in an inconsistent state.
  - **C**onsistency -
    - Ensures that the database is always in a valid state, regardless of the number of transactions performed on it.
    - Data is in a consistent state when a transaction starts and when it ends.
  - **I**solation -
    - Ensures that transactions do not interfere with each other if they are executing simultaneously.
    - Each transaction must be executed as if it were the only transaction in the system, to prevent conflicts and ensure data integrity
  - **D**urability -
    - Ensures that once a transaction is committed to the database, it will remain there even if the system fails or restarts.

        - This is achieved through data persistence mechanisms such as write-ahead logging, where changes are recorded in a log before they are applied to the database.

Advantages:

1. Data integrity: It enforces data integrity using its ACID properties, which ensures that data is always in a consistent and valid state.
2. Scalability: They can scale horizontally and vertically, allowing them to handle large volumes of data and high levels of concurrency.
3. Backup and recovery: It provides built-in backup and recovery mechanisms, which allow users to restore the database to a previous state in the event of a failure or disaster.
4. Data accessibility: These provide efficient and fast data access, even with large datasets.

Disadvantages

1. Limited Scalability: These can have difficulty handling extremely large datasets and may struggle to scale efficiently.
2. Lack of Flexibility: It is a highly structured language and may not be flexible enough to handle some types of data or data relationships.
3. Backup and Recovery: Backing up and recovering SQL databases can be a complicated process, requiring regular backups and the expertise to recover data in case of a disaster.
4. Performance Issues: SQL queries can be slow and resource-intensive, especially when working with large datasets or complex queries

NoSQL Databases

- NoSQL databases are non-relational databases that store data in a flexible and scalable manner.
- They can handle large amounts of data, provide high performance and scalability.

Advantages:

1. Scalability: NoSQL databases can handle large amounts of data and provide horizontal scalability, making it easy to scale out as needed
2. Flexibility: NoSQL databases can accommodate changes in data structure, making it easy to add new fields and data types
3. Performance: NoSQL databases can provide high performance for read-heavy and write-intensive applications

4. Cost-effective: NoSQL databases can be more cost-effective than traditional SQL databases for large scale projects

Disadvantages:

1. Not optimal for Multiple Updates - 2 nodes may have different data for the same property due to multiple updates in the database.
2. Not read optimised - It needs to search in the whole object to find the value we need.
3. Relations are not implicit - We cannot define the similarity between 2 objects as compared to in SQL databases where we can find common rows/columns.
4. Joins are hard - Joining 2 objects is hefty as it would run through each block and find similarities to join them.

Difference between SQL and NoSQL databases

SQL	NoSQL
1. Have fixed Schema	1. Have dynamic schema (JSON object)
2. Store data in structured tables	2. Store data as collection of Documents, key-value pairs etc.
3. Foreign key relationships are used to link data from different tables.	3. Data is contained in one block and there is no need for foreign key relationships.
4. Follows ACID properties	4. Follows CAP theorem
5. Data more consistent	5. Sacrifices consistency in order to achieve greater scalability and performance.
6. Used for applications that require complex queries and transactional consistency.	6. Used for applications that require high scalability and availability, such as real-time data processing.