A database is an organized collection of data that is stored and accessed electronically. It allows efficient storage, retrieval, and management of large volumes of information. Databases are commonly used in applications such as banking systems, university management systems, e-commerce platforms, and social media applications. To interact with a database, a special software known as a Database Management System (DBMS) is used. A DBMS provides an interface between users and the database, enabling data creation, modification, deletion, and retrieval while ensuring data security and integrity.

There are different types of databases based on how data is stored and managed. One of the most common types is the relational database, where data is organized into tables consisting of rows and columns. Each table represents an entity, and relationships between tables are established using keys. Relational databases follow a structured schema and use SQL (Structured Query Language) for data operations. Examples of relational database systems include MySQL, PostgreSQL, and Oracle Database. These databases are widely used in traditional enterprise applications.

Another important category is NoSQL databases, which are designed to handle large-scale, unstructured, or semi-structured data. Unlike relational databases, NoSQL databases do not rely on fixed schemas, making them highly flexible and scalable. They are commonly used in big data applications, real-time analytics, and distributed systems. NoSQL databases can be classified into document-based databases such as MongoDB, key-value stores like Redis, column-oriented databases such as Cassandra, and graph databases like Neo4j.

Database keys play a crucial role in identifying and maintaining relationships between records. A primary key is a unique identifier for each record in a table and cannot contain null values. It ensures that every row in a table can be uniquely distinguished from others. A foreign key is a field in one table that refers to the primary key of another table, helping maintain referential integrity between related tables. Candidate keys are attributes or sets of attributes that can uniquely identify records, out of which one is selected as the primary key.

Normalization is the process of organizing data in a database to minimize redundancy and avoid data anomalies. It involves dividing large tables into smaller, well-structured tables and defining relationships between them. The first normal form ensures that all attributes contain atomic values with no repeating groups. The second normal form eliminates partial dependency by ensuring that non-key attributes depend on the entire primary key. The third normal form removes transitive dependency so that non-key attributes depend only on the primary key. Normalization improves data consistency, reduces storage space, and simplifies maintenance.

Structured Query Language (SQL) is used to interact with relational databases. SQL commands are broadly divided into Data Definition Language (DDL) and Data Manipulation Language (DML). DDL commands such as CREATE, ALTER, and DROP are used to define and modify the structure of database objects like tables and schemas. DML commands including INSERT, UPDATE, DELETE, and SELECT are used to manipulate and retrieve data stored in the database.

ACID properties are fundamental principles that ensure reliable database transactions. Atomicity ensures that a transaction is treated as a single unit, meaning it either completes fully or does not execute at all. Consistency ensures that the database remains in a valid state before and after a transaction. Isolation ensures that concurrent transactions do not interfere with each other, and Durability guarantees that once a transaction is committed, its effects are permanently saved in the database even in the case of system failure.

A database schema defines the overall structure of a database, including tables, fields, relationships, constraints, and indexes. It acts as a blueprint that describes how data is organized and how different entities relate to each other. A well-designed schema improves performance, ensures data consistency, and makes the database easier to understand and maintain. Schemas are especially important in relational databases, where strict structure and relationships are enforced.

An entity represents a real-world object or concept that can be uniquely identified, such as a student, employee, or product. Each entity is described using attributes, which represent the properties of that entity. For example, a student entity may have attributes such as student ID, name, age, and department. Entities are usually mapped to tables in relational databases, where each row represents an instance of the entity and each column represents an attribute.

Relationships describe how entities are connected to one another. Common types of relationships include one-to-one, one-to-many, and many-to-many relationships. In a one-to-many relationship, a single record in one table can be associated with multiple records in another table, such as a department having many employees. Many-to-many relationships occur when multiple records in one table are associated with multiple records in another table and are typically implemented using a junction table.

Constraints are rules enforced on database columns to maintain data integrity. Common constraints include NOT NULL, UNIQUE, CHECK, and DEFAULT. The NOT NULL constraint ensures that a column cannot store null values, while the UNIQUE constraint ensures that all values in a column are distinct. CHECK constraints enforce specific conditions on data, such as restricting age values to a valid range. DEFAULT constraints assign a default value when no value is specified during insertion.

Indexes are database objects used to improve the speed of data retrieval operations. They work similarly to an index in a book, allowing the database to locate records quickly without scanning the entire table. While indexes significantly improve query performance, they also require additional storage space and can slow down insert, update, and delete operations. Therefore, indexes must be created carefully based on query usage patterns.

A transaction is a sequence of database operations that are treated as a single logical unit of work. Transactions are used to ensure data accuracy and consistency, especially in systems where multiple users access the database simultaneously. If any operation within a transaction fails, the entire transaction can be rolled back to its previous state. This ensures that incomplete or incorrect data is not stored in the database.

Concurrency control is the mechanism used to manage simultaneous access to the database by multiple users. Without proper concurrency control, issues such as dirty reads, lost updates, and phantom reads may occur. Techniques like locking, timestamps, and multiversion concurrency control are used to ensure that transactions execute correctly while maintaining database consistency and performance.

A view is a virtual table created using a SELECT query. Unlike physical tables, views do not store data themselves but display data derived from one or more tables. Views are commonly used to simplify complex queries, restrict access to sensitive data, and present customized data views to different users. Since views depend on underlying tables, any changes to the base tables are reflected automatically in the view.

Stored procedures are precompiled SQL statements stored within the database. They allow developers to encapsulate business logic at the database level, improving performance and security. Stored procedures reduce network traffic by executing multiple SQL statements in a single call and help enforce consistency across applications that access the database.

Triggers are special database objects that automatically execute predefined actions in response to certain events such as INSERT, UPDATE, or DELETE operations. Triggers are often used for auditing, enforcing complex constraints, and maintaining derived data. However, excessive use of triggers can make database behavior difficult to understand and debug.

Backup and recovery are critical aspects of database management. Database backups are copies of data stored separately to prevent data loss due to hardware failures, software crashes, or cyberattacks. Recovery techniques allow the database to be restored to a consistent state after failure. Regular backups and well-defined recovery strategies are essential for ensuring data availability and reliability.

Data redundancy refers to the unnecessary duplication of data within a database. While some redundancy may be intentional for performance optimization, excessive redundancy can lead to inconsistencies and increased storage costs. Normalization techniques are commonly applied to minimize redundancy by ensuring that each piece of data is stored only once. However, in certain scenarios such as data warehousing, controlled redundancy is allowed to improve query performance.

Data integrity ensures the accuracy, consistency, and reliability of data stored in a database throughout its lifecycle. Integrity is maintained using constraints, rules, and relationships defined in the database schema. Entity integrity ensures that each table has a primary key and that it uniquely identifies records. Referential integrity ensures that foreign key values correspond to existing primary key values in related tables. Domain integrity restricts the type, format, and range of values allowed in a column.

A data model is a conceptual representation of how data is structured and related within a database system. Common data models include the hierarchical model, network model, relational model, and object-oriented model. The relational model is the most widely used due to its simplicity and mathematical foundation based on set theory and predicate logic. Data models help in understanding system requirements and guide the design of databases.

Entity-Relationship (ER) modeling is a technique used to visually represent database structure during the design phase. ER diagrams consist of entities, attributes, and relationships. Attributes can be classified as simple, composite, multivalued, or derived. ER modeling helps identify entities and their relationships before converting them into relational tables, reducing design errors and improving clarity.

Denormalization is the process of intentionally introducing redundancy into a database to improve performance. It is often used in read-heavy applications where complex joins may slow down query execution. Denormalization must be carefully managed to avoid data inconsistencies, as updates may need to be applied in multiple places. It is commonly used in data warehouses and reporting systems.

A data warehouse is a centralized repository designed for analytical processing rather than transaction processing. It stores historical data collected from multiple sources and supports complex queries and reporting. Data warehouses use techniques such as ETL (Extract, Transform, Load) to clean and transform data before storage. Unlike OLTP systems, data warehouses are optimized for read operations and large-scale data analysis.

Online Transaction Processing (OLTP) systems are designed to handle a large number of short, real-time transactions such as insertions, updates, and deletions. These systems emphasize data consistency, fast response times, and concurrency control. Examples include banking systems, reservation systems, and retail point-of-sale systems. OLTP databases are typically highly normalized to ensure efficient transaction processing.

Database security is a critical concern in modern systems due to increasing cyber threats and regulatory requirements. Security measures include authentication, authorization, encryption, and auditing. Authentication verifies the identity of users, while authorization determines what actions they are allowed to perform. Encryption protects sensitive data both at rest and in transit, ensuring confidentiality even if unauthorized access occurs.

Access control mechanisms define how users interact with database resources. Role-based access control assigns permissions based on user roles rather than individual users, simplifying administration. Privileges such as SELECT, INSERT, UPDATE, and DELETE can be granted or revoked to control access. Proper access control helps prevent unauthorized data modification and leakage.

Data backup strategies include full backups, incremental backups, and differential backups. Full backups copy the entire database, while incremental backups store only changes made since the last backup. Differential backups store changes since the last full backup. Choosing the right backup strategy depends on recovery requirements, storage availability, and system performance.

Query processing refers to the steps taken by a database management system to execute a user's SQL query efficiently. When a query is submitted, the DBMS first parses and validates it to ensure correct syntax and semantics. The query is then translated into an internal representation and passed to the query optimizer. The optimizer evaluates multiple execution plans and selects the most efficient one based on factors such as indexing, data distribution, and cost estimation. Efficient query processing is critical for high-performance database systems.

Query optimization is the process of selecting the best execution strategy among multiple alternatives. The database optimizer uses statistics about tables, indexes, and data distribution to estimate the cost of different plans. Optimization techniques include choosing appropriate join orders, selecting efficient join algorithms, and using indexes effectively. Cost-based optimization is commonly used in modern databases to minimize response time and resource consumption.

Indexes play a vital role in query optimization by reducing the amount of data scanned during query execution. Common types of indexes include B-tree indexes, hash indexes, and bitmap indexes. B-tree indexes are widely used due to their balanced structure and support for range queries. Hash indexes provide fast equality-based searches but are not suitable for range queries. Bitmap indexes are effective in data warehousing environments where columns have low cardinality.

Joins are operations used to combine data from multiple tables based on a related column. Common join types include inner join, left outer join, right outer join, and full outer join. An inner join returns only matching records from both tables, while outer joins include unmatched records from one or both tables. Join algorithms such as nested loop join, hash join, and sort-merge join are selected by the optimizer based on data size and indexing.

A distributed database is a collection of data stored across multiple physical locations but managed as a single logical database. Distributed databases improve availability, scalability, and fault tolerance. Data may be distributed using techniques such as replication and fragmentation. Replication stores copies of data at multiple locations to improve availability, while fragmentation divides data into smaller parts and distributes them across nodes.

The CAP theorem is a fundamental concept in distributed databases that states a system can guarantee only two of the following three properties at the same time: consistency, availability, and partition tolerance. Consistency ensures all nodes see the same data at the same time, availability ensures the system responds to every request, and partition tolerance ensures the system continues to operate despite network failures. Different distributed databases prioritize different combinations of these properties based on application needs.

Replication strategies determine how data copies are maintained across distributed systems. In synchronous replication, data is written to all replicas before a transaction is committed, ensuring strong consistency but higher latency. In asynchronous replication, data is written to replicas after the transaction commits, improving performance but allowing temporary inconsistencies. Replication enhances fault tolerance and read scalability.

Sharding is a technique used to horizontally partition data across multiple servers. Each shard contains a subset of the data, allowing the database to scale by distributing load. Sharding improves performance for large-scale applications but increases complexity in query execution and transaction management. Proper shard key selection is essential to ensure balanced data distribution.

Cloud databases are databases hosted on cloud platforms and offered as managed services. They provide advantages such as automatic scaling, high availability, and reduced administrative overhead. Cloud databases can be relational or NoSQL and support pay-as-you-go pricing models. Popular use cases include web applications, analytics, and global-scale systems.

Data consistency models define how updates are propagated across a database system. Strong consistency ensures that all users see the latest data immediately after an update. Eventual consistency allows temporary inconsistencies but guarantees that all replicas will converge to the same value over time. Eventual consistency is commonly used in large-scale distributed systems to improve performance and availability.