

Important Elements

EQUATIONS & FORMULAS:

- Data = collection of raw, unorganized facts
- Information = processed, organized data providing context for decision-making
- Data vs Information: Data is raw and unorganized, while information is organized and meaningful
- Database = electronic place/system for storing, managing, and updating data
- DBMS = collection of interrelated data and programs for accessing and managing data
- Three Schema Architecture: Physical level, Logical level, View level
- ER Model: Entity, Attributes, Relationships, Specialisation, Generalisation

KEY CONCEPTS:

- Data: Raw, unorganized facts without specific purpose or significance
- Information: Processed, organized data providing context for decision-making
- Database: Electronic system for storing, managing, and updating data
- DBMS: Collection of data and programs for accessing and managing data efficiently
- Three Schema Architecture: Provides users with different levels of abstraction for interacting with data
- ER Model: High-level data model based on entities, attributes, and relationships
- Specialisation: Subgrouping an entity set into distinct sub entity sets based on functionality
- Generalisation: Reverse of specialisation, combining sub entity sets back into a parent entity set

DIAGRAMS & FLOWCHARTS:

- ER Diagram: Graphical representation of entities, attributes, and relationships in the ER Model
- Three Schema Architecture: Visual representation of the physical, logical, and view levels of data abstraction
- Specialisation Diagram: Shows how a parent entity set is divided into sub entity sets based on functionality
- Generalisation Diagram: Illustrates how sub entity sets are combined back into a parent entity set

EQUATIONS & FORMULAS:

- Super Key (SK): Any P&C; of attributes present in a table which can uniquely identify each tuple.
- Candidate Key (CK): Minimum subset of super keys, which can uniquely identify each tuple. It contains no redundant attribute.
- Primary Key (PK): Selected out of CK set, has the least number of attributes.
- Alternate Key (AK): All CK except PK.
- Foreign Key (FK): Creates a relationship between two tables.
- Composite Key: PK formed using at least 2 attributes.
- Compound Key: PK formed using 2 FK.
- Surrogate Key: Synthetic PK generated automatically by DB, usually an integer value.

KEY CONCEPTS:

- Specialisation & Generalisation:
- Specialisation is used to group entities with unique attributes, refining the database design.
- Generalisation is the reverse process of specialisation, creating a super class for entities with overlapping attributes.
- Attribute Inheritance:

- Both specialisation and generalisation involve attribute inheritance, where higher-level entity set attributes are inherited by lower-level entity sets.
- Participation Inheritance:
 - If a parent entity set participates in a relationship, its child entity sets will also participate in that relationship.
- Aggregation:
 - Aggregation is a technique to show relationships among relationships, treating them as higher-level entities to avoid redundancy.

DIAGRAMS & FLOWCHARTS:

- The ER diagram to Relational Model conversion process involves:
 - Strong Entity: Becomes an individual table with entity name and attributes as columns.
 - Weak Entity: Forms a table with all attributes and includes the PK of the corresponding Strong Entity as FK.
 - Single Value Attributes: Represented directly as columns in tables/relations.
 - Composite Attributes: Handled by creating separate attributes in the original relation for each composite attribute.
 - Multivalued Attributes: Creates new tables for each multivalued attribute, using the PK of the entity as a column FK in the new table.
 - Derived Attributes: Not considered in the tables.
- Generalisation:
 - Method-1 involves creating a table for the higher-level entity set and separate tables for lower-level entity sets.
 - Method-2 is an alternative representation for disjoint and complete generalisation scenarios, avoiding duplication of values.
- Aggregation:
 - Involves creating a table for the relationship set with attributes including primary keys of entity sets and aggregation set's entities.

EQUATIONS & FORMULAS:

1. Functional Dependency (FD):

- $X \rightarrow Y$
- X is the determinant, Y is the dependent

2. Armstrong's axioms for FD:

- Reflexive: $A \supseteq B$ then $A \rightarrow B$
- Augmentation: If $A \rightarrow B$ then $AX \rightarrow BX$
- Transitivity: If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

3. Normal forms:

- 1NF: Ensures atomic values in every cell
- 2NF: No partial dependency, non-prime attributes fully dependent on PK
- 3NF: No transitivity dependency, non-prime attributes should not depend on other non-prime attributes

KEY CONCEPTS:

- WHERE clause:
 - Used to filter rows based on specified conditions
- GROUP BY:
 - Used to group data based on one or more columns
- JOIN:
 - Combines data from multiple tables based on matching values
- Subqueries:
 - Nested queries used in WHERE, FROM, or SELECT clauses

- Database Normalization:
- Process of organizing a database to reduce redundancy and anomalies
- Constraints (DDL):
- Primary Key, Foreign Key, Unique, Check, Default
- Data Manipulation Language (DML):
- INSERT, UPDATE, DELETE operations in SQL
- Views:
- Virtual tables created by queries on other tables
- Set Operations:
- UNION, INTERSECT, MINUS to combine multiple SELECT statements

DIAGRAMS & FLOWCHARTS:

- A diagram illustrating how the WHERE clause filters rows based on conditions in a SQL query, with examples of AND, OR, and NOT conditions.
- A flowchart showing the process of GROUP BY clause grouping data based on specified columns and how it is used with aggregation functions.
- An illustration of the process of JOIN operations in SQL, showcasing INNER, LEFT, RIGHT, FULL, CROSS, and SELF JOIN, with examples of each type.
- A visual representation of how subqueries are embedded within SQL queries in WHERE, FROM, or SELECT clauses, with examples of different types of subqueries and their usage.
- A flowchart demonstrating the steps involved in the normalization process, from identifying functional dependencies to reaching different normal forms (1NF, 2NF, 3NF).

EQUATIONS & FORMULAS:

- 1NF: Every relation cell must have atomic value. No multi-valued attributes allowed.
- 2NF: Non-prime attributes must be fully dependent on the primary key. No partial dependency allowed.

- 3NF: No transitivity dependency exists. Non-prime attribute should not depend on another non-prime attribute.

- BCNF: Functional dependency $A \rightarrow B$, A must be a super key.

KEY CONCEPTS:

- Normal Forms:

- 1NF: Atomic values, no multi-valued attributes.

- 2NF: No partial dependency, non-prime attributes fully dependent on PK.

- 3NF: No transitivity dependency.

- BCNF: A must be a super key in functional dependency $A \rightarrow B$.

- Advantages of Normalization:

- Minimize data redundancy.

- Improve overall database organization.

- Maintain data consistency.

- Transactions:

- Unit of work in a logical sequence against the database.

- ACID properties: Atomicity, Consistency, Isolation, Durability.

- Transaction states: Active, Partially committed, Committed, Failed, Aborted, Terminated.

- Atomicity and Durability in Transactions:

- Recovery mechanisms ensure atomicity and durability.

- Shadow-copy scheme for atomicity.

- Log-based recovery methods for durability.

- Indexing in DBMS:

- Optimize database performance by minimizing disk accesses.
- Types of indexing: Primary (clustering), Secondary (non-clustering).
- Advantages of indexing: Faster data access, less IO.
- Limitations of indexing: Additional space, decreased performance in certain queries.
- NoSQL:
- Non-tabular databases storing data differently than relational tables.
- Types: document, key-value, wide-column, graph.
- Advantages: Flexible schema, horizontal scaling, high availability, easy insert and read operations.
- Use cases: Fast-paced Agile development, huge volumes of data, scale-out architecture.

DIAGRAMS & FLOWCHARTS:

- No specific diagrams or flowcharts were mentioned in the text.

EQUATIONS & FORMULAS:

- No equations or formulas provided in the text.

KEY CONCEPTS:

- NoSQL databases: Non-tabular databases that store data differently than relational tables, offering flexible schemas and scalability with large data and high user loads.
- Types of NoSQL data models: Key-Value Stores, Column-Oriented, Document-Based Stores, and Graph-Based Stores.
- MongoDB: A NoSQL database that supports ACID transactions.
- NoSQL misconceptions: NoSQL databases can store relationship data efficiently and support ACID transactions.
- Hierarchical Databases: Data model appropriate for concrete hierarchy-based information gathering.

- Network Databases: Extension of hierarchical databases allowing child records to associate with multiple parent records.
- Database Clustering: Process of combining multiple servers or instances connecting a single database for data redundancy and load balancing.

DIAGRAMS & FLOWCHARTS:

- No diagrams or flowcharts provided in the text.

EQUATIONS & FORMULAS:

- High availability = amount of time a database is considered available
- Availability needed depends on number of transactions and frequency of analytics
- Parallelism, availability, performance, manageability, and cost reduction are advantages of partitioning
- Sharding = technique to implement horizontal partitioning
- Cons of sharding: complexity, non-uniformity, not suited for analytical queries

KEY CONCEPTS:

- Load balancing: distributing incoming network traffic across multiple servers to prevent overload on a single server
- High availability: ensuring database access is available for a certain amount of time, crucial for uninterrupted service
- Clustering: splitting requests among multiple computers for better load balancing and high availability
- Partitioning: dividing database objects into separate servers for increased performance and manageability
- Vertical partitioning: slicing relation vertically/column-wise
- Horizontal partitioning: slicing relation horizontally/row-wise
- Distributed database: single logical database spread across multiple locations and interconnected by a network

- Sharding: splitting data across multiple DB instances and introducing a routing layer for scalability and availability

DIAGRAMS & FLOWCHARTS:

- Diagram needed to illustrate how load balancing distributes traffic among multiple servers
- Flowchart showing the process of clustering requests among multiple computers for load balancing and high availability
- Flowchart demonstrating the steps involved in partitioning a database into separate servers for increased performance and manageability
- Diagram showing how sharding splits data across different DB instances and utilizes a routing layer to forward requests to the correct instances.

EQUATIONS & FORMULAS:

- CAP Theorem: Consistency, Availability, Partition Tolerance
- CA Databases: Consistency, Availability
- CP Databases: Consistency, Partition Tolerance
- AP Databases: Availability, Partition Tolerance

KEY CONCEPTS:

- Database Scaling Patterns:
- Query Optimisation & Connection Pool Implementation
- Vertical Scaling or Scale-up
- Command Query Responsibility Segregation (CQRS)
- Multi Primary Replication
- Partitioning of Data by Functionality
- Horizontal Scaling or Scale-out
- Data Centre Wise Partition

- CAP Theorem:
- Consistency: All nodes see the same data simultaneously
- Availability: System remains operational all the time
- Partition Tolerance: System does not fail despite communication breaks
- Master-Slave Database Concept:
- Master holds true or latest data for write operations
- Slaves used for reading operations
- DB replication distributes data from Master to Slaves

DIAGRAMS & FLOWCHARTS:

- CAP Theorem diagram:
- Components: Consistency, Availability, Partition Tolerance
- Significance: Shows the tradeoff between consistency and availability in a distributed system
- Master-Slave Database Architecture:
- Components: Master DB, Slave DBs
- Significance: Illustrates the distribution of read and write operations in a high-traffic system to optimize IO and reduce latency