

Comprehensive Summary

Study Summary:

1. Data Concepts:

- Data: Collection of raw, unorganized facts without specific purpose
- Types of Data: Quantitative (numerical) and Qualitative (descriptive)
- Information: Processed, organized data that enables decision-making
- Data vs Information: Data is raw, while information provides context
- Database: Electronic system for storing, managing, and accessing data
- DBMS: Collection of data and programs for data access and management

2. DBMS Architecture:

- Three Schema Architecture: Physical, Logical, and View levels
- Instances and Schemas: Collection of data at a moment vs overall design
- Data Models: Description of database design at the logical level
- Database Languages: DDL for schema, DML for queries and updates
- Database Access: Applications interact with DB using host languages
- Database Administrator: Controls data and program access

3. Entity-Relationship Model:

- Entities: Objects in the real world with physical existence and unique identification
- Attributes: Characteristics of entities, simple, composite, single-valued, multi-valued, derived
- Relationships: Associations between entities, strong vs weak relationships
- ER Notations: Representing entities, attributes, and relationships

4. Extended ER Features:

- Specialisation: Subgrouping entities into distinct subgroups based on functionalities
- Generalisation: Reverse of specialisation, grouping entities under a common superclass

Key Concepts:

- Data and Information distinction
- Database Management Systems (DBMS) and its components
- Three Schema Architecture in DBMS
- Entity-Relationship Model for database design
- Specialisation and Generalisation in Extended ER Features

Note: This summary covers the fundamental concepts of DBMS, database architecture, and entity-relationship modeling, providing a comprehensive overview for exam preparation.- Specialisation:

- "is-a" relationship between superclass and subclass
- Depicted by a triangle component
- Reasons for specialisation:
 - Certain attributes may only be applicable to a few entities of the parent entity set
 - Helps to show the distinctive features of sub entities
 - To group entities and refine the database blueprint
- Generalisation:
 - Reverse of specialisation
 - DB designer may create a new generalised entity set when properties of two entities overlap
- "is-a" relationship between subclass and superclass

- Example: Car, Jeep, and Bus can be generalised to a new entity set "Vehicle"
- Bottom-up approach
- Reasons for generalisation:
 - Makes the database more refined and simpler
 - Avoids repeating common attributes
- Attribute Inheritance:
 - Present in both specialisation and generalisation
 - Attributes of higher level entity sets are inherited by lower level entity sets
 - Example: Customer & Employee inherit attributes of Person
- Participation Inheritance:
 - If a parent entity set participates in a relationship, its child entity sets will also participate in that relationship
- Aggregation:
 - Technique to show relationships among relationships
 - Abstraction applied to treat relationships as higher-level entities
 - Helps to avoid redundancy by aggregating relationships as entity sets themselves
- Relational Model (RM):
 - Organises data in the form of relations (tables)
 - Each table has a unique name and represents a relationship among a set of values
 - Tuple represents a single row or unique record in a table
 - Columns represent attributes of the relation, each with a permitted value domain
 - Relational DBMS systems include Oracle, IBM, MySQL, and MS Access

- Relational Model Keys:

- Super Key (SK), Candidate Key (CK), Primary Key (PK), Alternate Key (AK), Foreign Key (FK), Composite Key, Compound Key, Surrogate Key

- Integrity Constraints:

- Ensure data consistency and prevent corruption in the database

- Domain Constraints restrict attribute values, Entity Constraints require a primary key, Referential Constraints maintain consistency between tuples in different relations

- Transforming ER Model to Relational Model:

- Strong Entity becomes an individual table, Weak Entity forms a table with attributes and a composite primary key

- Single and Multivalued Attributes represented in tables

- Derived Attributes not considered

- Generalisation methods for creating tables based on higher and lower level entity sets

- SQL:

- Structured Query Language used for accessing and manipulating data

- CRUD operations: CREATE, READ, UPDATE, DELETE

- RDBMS: Relational Database Management System enables implementation of the relational model

- SQL Data Types:

- Data stored in tables with different types (e.g., CHAR, VARCHAR, INT, DATE)

- Data types have specific sizes and characteristics

- SQL Commands:

- DDL, DRL/DQL, DML, DCL, TCL for defining schema, retrieving data, modifying data, controlling access, and managing transactions

- Data Retrieval Language (DRL):
- Syntax, WHERE, BETWEEN, IN, AND/OR/NOT, IS NULL for querying data
- Managing DB (DDL):
- Creation, selection, dropping, and listing of databases and tables.- ****WHERE Clause****:
- Used to reduce rows based on given conditions
- Examples:
- ``SELECT * FROM customer WHERE age > 18;``
- ****BETWEEN Clause****:
- Selects data within a range, inclusive of the specified values
- Example: ``SELECT * FROM customer WHERE age between 0 AND 100;``
- ****IN Clause****:
- Reduces OR conditions by specifying a list of values
- Example: ``SELECT * FROM officers WHERE officer_name IN ('Lakshay', 'Maharana Pratap', 'Deepika');``
- ****AND/OR/NOT****:
- Logical operators for combining conditions in WHERE clauses
- ``AND``: Both conditions must be true
- ``OR``: At least one condition must be true
- ``NOT``: Excludes specified values
- ****IS NULL****:
- Filters data where a specific column is NULL
- Example: ``SELECT * FROM customer WHERE prime_status is NULL;``

- **Pattern Searching / Wildcard**:
- Use ` % ` for any number of characters and ` _ ` for a single character
- Example: `SELECT * FROM customer WHERE name LIKE '%p_';`
- **ORDER BY**:
- Sorts the retrieved data based on specified columns in ascending or descending order
- Example: `SELECT * FROM customer ORDER BY name DESC;`
- **GROUP BY**:
- Groups data based on one or more columns
- Used with aggregation functions like COUNT, SUM, AVG, MIN, MAX
- **DISTINCT**:
- Retrieves unique values from a table
- Example: `SELECT DISTINCT(col_name) FROM table_name;`
- **GROUP BY HAVING**:
- Filters grouped data based on specified conditions
- Similar to WHERE but used after GROUP BY
- **Constraints (DDL)**:
- Primary Key, Foreign Key, Unique, Check, Default constraints for data integrity
- **ALTER OPERATIONS**:
- Add, modify, change, drop columns, or rename tables
- **Data Manipulation Language (DML)**:
- INSERT, UPDATE, DELETE operations with examples

- **JOINING TABLES**:

- INNER JOIN, OUTER JOIN types (LEFT, RIGHT, FULL), CROSS JOIN, SELF JOIN

- **SET OPERATIONS**:

- UNION, INTERSECT, MINUS for combining select statements

- **SUBQUERIES**:

- Nested queries used within SELECT, FROM, or WHERE clauses

- **MySQL VIEWS**:

- Virtual tables based on base tables, no data stored

- **LEC-11: Normalisation**:

- Process to reduce redundancy and anomalies in database tables

- **Types of Normal forms**:

- 1NF, 2NF, 3NF to ensure data integrity and minimize redundancy.- **Types of Normal forms.**

1. **1NF**

- Every relation cell must have atomic value.

- Relation must not have multi-valued attributes.

2. **2NF**

- Relation must be in 1NF.

- No partial dependency.

- All non-prime attributes must be fully dependent on PK.

- Non-prime attribute cannot depend on part of the PK.

3. **3NF**

- Relation must be in 2NF.
- No transitivity dependency exists.
- Non-prime attribute should not find a non-prime attribute.

4. **BCNF (Boyce-Codd normal form)**

- Relation must be in 3NF.
- FD: $A \rightarrow B$, A must be a super key.
- Prime attribute should not be derived from any prime or non-prime attribute.
- **Advantages of Normalisation:**

1. Helps to minimise data redundancy.
2. Greater overall database organisation.
3. Data consistency is maintained in the database.

- **Transaction:**

1. **ACID Properties:**

- Atomicity: All operations of transaction are reflected properly in the DB, or none are.
- Consistency: Integrity constraints must be maintained before and after the transaction.
- Isolation: Multiple transactions can happen in isolation without interfering with each other.
- Durability: Changes made by a transaction persist even after system failures.

- **Transaction states:**

1. **Active state:**

- Initial state where read and write operations are performed.

2. **Partially committed state:**

- Changes are saved in the buffer in main memory.

3. ****Committed state:****

- Updates are made permanent on the DB.

4. ****Failed state:****

- Error occurs during execution, impossible to continue.

5. ****Aborted state:****

- Changes are reversed, transaction is rolled back.

6. ****Terminated state:****

- Transaction is committed or aborted.

- ****Implementing Atomicity and Durability in Transactions:****

1. ****Recovery Mechanism:****

- Supports atomicity and durability.

2. ****Shadow-copy scheme:****

- Copies of DB are made for updates.

- If transaction fails, new copy is deleted.

- Atomicity is ensured by maintaining the old content of DB.

3. ****Log-based recovery methods:****

- Sequence of records maintained for each transaction.

- Logs stored before actual transaction is applied.

- Immediate and deferred DB modifications for atomicity.

- ****Indexing in DBMS:****

1. ****Indexing Methods:****

- Primary Index (Clustering Index) and Secondary Index (Non-Clustering Index).
- Primary Index based on data file sorting w.r.t primary key or non-key attributes.
- Dense and Sparse Indices for optimising performance.

2. ****Advantages of Indexing:****

- Faster access and retrieval of data.

3. ****Limitations of Indexing:****

- Additional space required for index table.
- Decreased performance in INSERT, DELETE, and UPDATE queries.

- ****NoSQL:****

1. ****Advantages and History:****

- Schema-free, flexible data structures, scalability.
- Emerged due to decreasing storage costs and need for developer productivity.

2. ****NoSQL Databases Advantages:****

- Flexible schema, horizontal scaling, high availability, easy insert and read operations, caching mechanism.

3. ****When to use NoSQL:****

- Agile development, structured and semi-structured data storage, huge data volumes, scale-out architecture requirements.- NoSQL databases store data differently than relational databases and come in various types including document, key-value, wide-column, and graph.
- NoSQL databases provide flexible schemas and scale easily with large amounts of data and high user loads.
- NoSQL databases are schema-free and can handle huge amounts of data (big data).
- Types of NoSQL data models include key-value stores, column-oriented stores, document-based stores, and graph-based stores.

- Key-value stores store data as key-value pairs and are ideal for systems that need to find and retrieve data quickly.

- Column-oriented stores organize data as a set of columns and are efficient for analytics and aggregating values.

- Document-based stores store data in documents similar to JSON objects and are suitable for e-commerce platforms and mobile app development.

- Graph-based stores focus on the relationship between data elements and are optimized for analyzing relationships in data sets.

- NoSQL databases are suitable for fast-paced Agile development, storage of structured and semi-structured data, huge volumes of data, scale-out architecture, and modern application paradigms like micro-services and real-time streaming.

- NoSQL databases can store relationship data differently than relational databases, with many finding it easier to model relationship data in NoSQL databases.

- NoSQL databases like MongoDB can support ACID transactions, contrary to common misconceptions.

- NoSQL databases have disadvantages such as data redundancy, costly update and delete operations, and limitations in fulfilling all application needs.

- SQL databases and NoSQL databases differ in data storage model, development history, examples, primary purpose, schemas, scaling, and ACID properties.

- Relational databases store information in tables with fixed rows and columns and are optimized for working with structured data.

- Object-oriented databases are based on the object-oriented programming paradigm and store data as objects, providing easy and quick data storage and retrieval.

- Hierarchical databases are suitable for scenarios where information gathering is based on a concrete hierarchy and have advantages in ease of use but disadvantages in inflexibility.

- Database clustering involves combining multiple servers or instances connecting a single database for data redundancy and load balancing.- Load Balancing:

- Distributes traffic across multiple machines

- Prevents overworking of individual machines

- Ensures seamless scaling as traffic increases
- Essential for high availability
- High Availability:
 - Refers to the amount of time a database is considered available
 - Depends on transaction volume and analytics frequency
 - Achieved through database clustering and load balancing
 - Allows for continuous availability even if a server shuts down
- Clustering:
 - Requests are split among multiple computers
 - Ensures load balancing and high availability
 - Reduces risk of system failures
 - Involves handling requests by other nodes if one node fails
- Partitioning:
 - Divides a large database into smaller partitions
 - Enables easier management of large database tables
 - Improves performance and scalability
 - Vertical partitioning slices data column-wise, while horizontal partitioning slices data row-wise
- Advantages of Partitioning:
 - Parallelism, availability, performance, manageability
 - Reduces costs compared to vertical scaling
- Distributed Database:

- Single logical database spread across multiple servers
- Linked by a network
- Result of clustering, partitioning, and sharding
- Sharding:
 - Implements horizontal partitioning
 - Splits data across multiple instances for scalability
 - Pros include scalability and availability
 - Cons include complexity and suitability for analytical queries- **Concepts:**
- CAP Theorem: Consistency, Availability, Partition Tolerance
- NoSQL Databases: CA, CP, AP
- Master-Slave Database Concept
- **CAP Theorem:**
 - Consistency: All nodes see the same data simultaneously.
 - Availability: System remains operational all the time.
 - Partition Tolerance: System does not fail in case of communication break between nodes.
- Theorem states a distributed system can only provide two of three properties simultaneously.
- **NoSQL Databases:**
 - CA Databases: Consistency, Availability (MySQL, PostgreSQL)
 - CP Databases: Consistency, Partition Tolerance (MongoDB)
 - AP Databases: Availability, Partition Tolerance (Cassandra, DynamoDB)
- **Master-Slave Database Concept:**

- Optimizes IO in high-traffic systems.
- Master stores latest data, directs write operations.
- Slaves used for reading operations.
- DB replication distributes data from Master to Slaves synchronously or asynchronously.
- ****Scaling Patterns:****
- Query Optimisation & Connection Pool Implementation
- Vertical Scaling (Scale-up)
- CQRS (Command Query Responsibility Segregation)
- Multi Primary Replication
- Partitioning of Data by Functionality
- Horizontal Scaling (Scale-out)
- Data Centre Wise Partition
- ****Case Study: Cab Booking APP****
- Initial issues with scaling as requests increased.
- Solutions included query optimization, vertical scaling, CQRS, multi primary replication, partitioning of data, horizontal scaling, and data centre wise partitioning.