Chapter 02

Review Questions

2.1. Define the following terms: data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML, SDL, VDL, query language, host language, data sublanguage, database utility, catalog, client/server architecture, three-tier architecture, and n-tier architecture.

Answer :

Here are the terms with shorter definitions:

1. **Data Model** – Framework defining how data is structured and related.
2. **Database Schema** – The design or structure of the database, including tables and relationships.
3. **Database State** – The current data in the database at a specific time.
4. **Internal Schema** – The physical storage and organization of data on disk.
5. **Conceptual Schema** – The logical view of data, independent of physical storage.
6. **External Schema** – User-specific view of the data, defining what data is visible.
7. **Data Independence** – The ability to change one level of schema without affecting others.
8. **DDL (Data Definition Language)** – SQL commands for defining and modifying database structure.
9. **DML (Data Manipulation Language)** – SQL commands for querying and modifying data.
10. **SDL (Storage Definition Language)** – Defines how data is physically stored.
11. **VDL (View Definition Language)** – Defines user-specific views of the data.
12. **Query Language** – Language for querying and manipulating data (e.g., SQL).
13. **Host Language** – A programming language (like C or Java) that interacts with the database.
14. **Data Sublanguage** – A combination of DDL and DML used within a host language.
15. **Database Utility** – Tools for database management, like backups and performance tuning.
16. **Catalog** – Metadata repository storing information about the database structure.
17. **Client/Server Architecture** – Model where clients request services from a central server.
18. **Three-Tier Architecture** – Application model with three layers: presentation, logic, and data.
19. **N-Tier Architecture** – Extension of three-tier with additional modular layers.

2.2. Discuss the main categories of data models. What are the basic differences among the relational model, the object model, and the XML model?

Answer :

Main Categories of Data Models:

1. **Hierarchical Model** – Data organized in a tree structure with parent-child relationships.
2. **Network Model** – Similar to hierarchical but allows multiple parent relationships.
3. **Relational Model** – Data stored in tables with rows and columns, linked by foreign keys.
4. **Object-Oriented Model** – Data represented as objects with attributes and methods.
5. **Entity-Relationship Model** – Focuses on entities and relationships, used for database design.
6. **XML Model** – Data represented in a tree-like structure using XML tags, useful for data exchange.

Differences Among Relational, Object, and XML Models:

1. **Relational Model**
   - Data in tables with rows and columns.
   - Uses SQL for querying.
   - Best for structured data.
2. **Object-Oriented Model**
   - Data as objects with attributes and methods.
   - Supports complex relationships.
   - Suitable for complex, dynamic data.
3. **XML Model**
   - Data in a tree structure using XML tags.
   - Uses XPath or XQuery for querying.
   - Great for data exchange between systems but less efficient for relational queries.

2.3. What is the difference between a database schema and a database state?

Answer :

Database Schema and **Database State** are related but distinct concepts:

1. **Database Schema**:
   - Refers to the **structure** or design of the database.
   - Defines **tables**, **relationships**, **constraints**, and **data types**.
   - It is **static**, meaning it does not change frequently unless the database design is modified.
   - Example: Defines a `Users` table with columns `user_id`, `name`, `email`.

2. **Database State**:
    - Refers to the **current contents** of the database at a particular point in time.
    - It represents the actual **data** stored in the database (the values in the tables).
    - It is **dynamic** and can change frequently as data is inserted, updated, or deleted.
    - Example: After a user registers, the `Users` table might have a row with `user_id = 1`, `name = 'Alice'`, and `email = 'alice@example.com'`.

2.4. Describe the three-schema architecture. Why do we need mappings among schema levels? How do different schema definition languages support this architecture?

Answer :

Three-Schema Architecture

1. **Internal Schema** – Describes the physical storage of data (e.g., file structures, indexes).
2. **Conceptual Schema** – Describes the logical structure of the entire database (e.g., tables, relationships, constraints).
3. **External Schema** – Describes how users or applications view the data (e.g., customized views for different users).

Need for Mappings

- Mappings ensure **data independence**, allowing changes at one schema level (e.g., physical storage) without affecting others (e.g., user views or logical structure).

Schema Definition Languages

1. **DDL (Data Definition Language)** – Defines the conceptual schema and external schemas (views).
2. **SDL (Storage Definition Language)** – Defines the internal schema (physical storage).
3. **VDL (View Definition Language)** – Defines external schemas (user views).

Mappings between these schemas enable flexibility and independence in managing the database.

2.5. What is the difference between logical data independence and physical data independence? Which one is harder to achieve? Why?

Answer :

- **Logical Data Independence**: Ability to change the **conceptual schema** (logical structure) without affecting **user views** or applications.
- **Physical Data Independence**: Ability to change the **internal schema** (physical storage) without affecting the **conceptual schema** or **user views**.

Physical Data Independence is harder to achieve because it involves managing physical storage changes without impacting the logical design, which requires complex systems and performance considerations.

2.6. What is the difference between procedural and nonprocedural DMLs?

Answer :

Procedural DML (Data Manipulation Language)

- **Definition**: Requires the user to specify **how** to obtain the desired result (i.e., the steps or procedure to retrieve or modify data).
- **Example**: In a procedural DML, you might specify a loop to fetch rows from a table.
- **Example Language**: C, embedded SQL, or programming languages with DML functions.

Nonprocedural DML

- **Definition**: Allows the user to specify **what** data to retrieve or modify, without specifying the procedure or steps for achieving the result.
- **Example**: A user simply specifies the condition (e.g., "select all customers where age > 30"), and the system decides how to execute the query.
- **Example Language**: SQL (Structured Query Language).

2.7. Discuss the different types of user-friendly interfaces and the types of users who typically use each.

Answer :

Types of User-Friendly Interfaces:

1. **Graphical User Interface (GUI)**
   - **Users**: End-users, casual users (e.g., business analysts).
   - **Description**: Visual elements like icons and buttons for interaction.
2. **Command-Line Interface (CLI)**
   - **Users**: DBAs, developers, advanced users.

    - **Description**: Text-based commands for system interaction.
3. **Natural Language Interface (NLI)**
    - **Users**: Non-technical, casual users.
    - **Description**: Users interact using plain language (e.g., "Show me all employees").
4. **Form-Based Interface**
    - **Users**: End-users, business users.
    - **Description**: Structured forms for data entry and querying.
5. **Web-Based Interface**
    - **Users**: End-users, remote users.
    - **Description**: Browser-based interface for remote data access.

Each interface suits different user needs based on their technical expertise and tasks.

2.8. With what other computer system software does a DBMS interact?

Answer :

A **Database Management System (DBMS)** interacts with various types of computer system software, including:

1. **Operating System (OS)**
    - Manages hardware resources (CPU, memory, disk storage) for the DBMS.
    - Handles file storage, process management, and security (e.g., Windows, Linux).
2. **Application Software**
    - Business applications interact with the DBMS to store and retrieve data (e.g., ERP, CRM, e-commerce platforms).
3. **Web Servers**
    - Web applications use a DBMS to store and retrieve dynamic content (e.g., Apache, Nginx).
4. **Networking Software**
    - Enables database access over networks, ensuring secure communication (e.g., TCP/IP protocols, VPNs).
5. **Middleware**
    - Connects applications with the database, facilitating data exchange (e.g., ODBC, JDBC).
6. **Backup and Recovery Software**
    - Ensures data safety by handling automated backups and restorations.
7. **Security Software**
    - Protects the database from threats via authentication, encryption, and firewalls.

2.9. What is the difference between the two-tier and three-tier client/server architectures?

Answer :

Two-Tier vs. Three-Tier Architecture

1. **Two-Tier Architecture**
   - **Structure**: Client (UI + logic) directly communicates with the database.
   - **Example**: Desktop application with a direct database connection.
   - **Limitation**: Less scalable, performance issues with multiple clients.
2. **Three-Tier Architecture**
   - **Structure**: Adds a middle layer (Application Server) between client and database.
   - **Example**: Web app where the client interacts with an app server, which then communicates with the database.
   - **Advantage**: More scalable, secure, and better for web applications.

Three-tier is preferred for web apps due to better scalability and security.

2.10. Discuss some types of database utilities and tools and their functions.

Answer :

Types of Database Utilities and Their Functions

1. **Backup and Recovery Tools** – Create and restore database backups (e.g., Oracle RMAN, MySQL mysqldump).
2. **Performance Monitoring Tools** – Analyze and optimize query and system performance (e.g., SQL Profiler).
3. **ETL & Data Migration Tools** – Move and transform data between databases (e.g., Talend, Informatica).
4. **Security Tools** – Manage authentication, encryption, and access control (e.g., Active Directory).
5. **Database Repair & Maintenance Tools** – Detect and fix corruption, optimize storage (e.g., DBCC CHECKDB).
6. **Query Optimization Tools** – Improve query performance (e.g., Oracle SQL Tuning Advisor).

These tools help maintain database security, efficiency, and reliability.

2.11. What is the additional functionality incorporated in n-tier architecture (n . 3)?

Answer :

Additional Functionality in N-Tier Architecture (n > 3)

In **n-tier architecture** (beyond three tiers), additional layers provide enhanced **scalability, flexibility, and security**. Some common additional tiers include:

1. **Load Balancing Tier** – Distributes incoming traffic across multiple servers to improve performance and reliability.
2. **Business Logic Tier (Separated from Application Layer)** – Enhances modularity by isolating core business rules.
3. **Data Caching Tier** – Speeds up data access by storing frequently used data in memory (e.g., Redis, Memcached).
4. **Security Tier** – Implements authentication, authorization, and encryption for better data protection.
5. **Analytics & Reporting Tier** – Provides business intelligence and reporting capabilities.

Exercises

2.12. Think of different users for the database shown in Figure 1.2. What types of applications would each user need? To which user category would each belong, and what type of interface would each need?

Answer :

Different Users for the Database:

- **End Users:** Use user-friendly interfaces to interact with the data (e.g., customers, students).
- **Application Programmers:** Develop backend systems and APIs to interact with the database.
- **DBAs:** Maintain and optimize the database with management tools.
- **Database Designers:** Design the database structure with data modeling tools.
- **Data Analysts:** Analyze data and generate reports using tools like SQL

2.13. Choose a database application with which you are familiar. Design a schema and show a sample database for that application, using the notation of Fig-ures 1.2 and 2.1. What types of additional information and constraints .would you like to represent in the schema? Think of several users of your database, and design a view for each.

Answer :

Schema Design for a Database Application

Let's consider a database for a **Library Management System**:

- **Entities:**
  - `Book(Book_ID, Title, Author, Genre, ISBN)`
  - `Member(Member_ID, Name, Address, Phone_Number)`
  - `Loan(Loan_ID, Member_ID, Book_ID, Loan_Date, Return_Date)`
- **Sample Data:**
  - `Book`: (1, "The Great Gatsby", "F. Scott Fitzgerald", "Fiction", "9780743273565")
  - `Member`: (101, "John Doe", "123 Main St", "555-1234")
  - `Loan`: (1001, 101, 1, "2025-01-01", "2025-01-15")
- **Additional Information and Constraints:**
  - `ISBN` should be unique across all books.
  - The `Loan_Date` and `Return_Date` must be valid dates, and the return date cannot precede the loan date.
  - A `Member` can borrow multiple books, but each loan must be unique in terms of `Loan_ID`.
- **User Views:**
  - **Librarian View:** Can access detailed information about books, members, and loans, with the ability to add or modify records.
  - **Member View:** Can only view the status of their loans and borrow new books.
  - **Admin View:** Can view and manage all entities in the system, including setting loan durations and adding new members.

2.14. If you were designing a Web-based system to make airline reservations and sell airline tickets, which DBMS architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

Answer :

DBMS Architecture for a Web-based Airline Reservation System

For a Web-based airline reservation and ticket sales system, I would choose a **Client-Server Architecture (2-Tier or 3-Tier)** because:

- **Client-Server (3-Tier)** is the best choice for scalability, performance, and security in a Web application. The three layers include:
  1. **Client Layer:** Users interact with the Web interface for making reservations and checking flight status.
  2. **Application Layer:** The middle tier handles business logic, manages bookings, and processes transactions.
  3. **Database Layer:** Stores flight schedules, customer data, reservations, and transactions.

- **Other architectures** like the **Flat DBMS architecture** (single-tier) would not be scalable or efficient for handling large numbers of users simultaneously. The **Distributed DBMS** could also be an option but is more complex to maintain and configure.

2.15. Consider Figure 2.1. In addition to constraints relating the values of col- umns in one table to columns in another table, there are also constraints that impose restrictions on values in a column or a combination of columns within a table. One such constraint dictates that a column or a group of columns must be unique across all rows in the table. For example, in the STUDENT table, the Student_number column must be unique (to prevent two different students from having the same Student_number). Identify the col- umn or the group of columns in the other tables that must be unique across all rows in the table.

Answer :

Identifying Unique Constraints in Database Tables

Here are some columns that must be unique across all rows in their respective tables:

- **STUDENT Table:**
  - `Student_number` should be unique to prevent two students from having the same identifier.
- **BOOK Table:**
  - `ISBN` should be unique to ensure no two books share the same ISBN.
- **LOAN Table:**
  - `Loan_ID` should be unique to distinguish each loan transaction.
- **MEMBER Table:**
  - `Member_ID` should be unique to ensure each member is identified by a distinct number.

Each of these constraints is crucial for maintaining data integrity in the system.