Chapter 01 (Ramzi Book)

Review Questions

1.1. Define the following terms: data, database, DBMS, database system, database catalog, program-data independence, user view, DBA, end user, canned transaction, deductive database system, persistent object, meta-data, and transaction-processing application.

Answer:

- **Data**: Raw facts and figures without context (e.g., numbers, names).
- **Database**: A structured collection of related data stored electronically.
- **DBMS (Database Management System)**: Software that manages databases, enabling data storage, retrieval, and security.
- **Database System**: A combination of a database, DBMS, and related applications.
- **Database Catalog**: Stores metadata, such as table structures and constraints.
- **Program-Data Independence**: Separation of data structure from application programs, allowing easier updates.
- **User View**: A customized representation of the database for different users.
- **DBA (Database Administrator)**: Manages database security, backup, and performance.
- **End User**: Individuals who interact with the database for various tasks.
- **Canned Transaction**: Predefined queries or transactions executed by users.
- **Deductive Database System**: A database that supports inference and logic-based queries.
- **Persistent Object**: Data that remains stored beyond the execution of a program.
- **Metadata**: Data that describes other data (e.g., table schemas, constraints).
- **Transaction-Processing Application**: Manages high-volume operations, such as banking transactions.

1.2. What four main types of actions involve databases? Briefly discuss each.

Answer:

Four Main Types of Database Actions

1. **Data Definition** – Creating/modifying database structures (e.g., tables, indexes).
2. **Data Manipulation** – Inserting, updating, deleting, and querying data.
3. **Data Retrieval** – Fetching specific data using queries.
4. **Database Administration** – Managing security, performance, and backup

1.3. Discuss the main characteristics of the database approach and how it differs from traditional file systems.

Answer:

Characteristics of the Database Approach vs. Traditional File Systems

- **Centralized Control** – Unlike file systems, databases centralize data management, reducing redundancy.
- **Data Independence** – Database schemas separate data from applications, unlike file systems.
- **Efficient Querying** – Databases use optimized query languages (e.g., SQL), unlike manual file processing.
- **Concurrent Access** – Multiple users can access a database safely, whereas file systems may cause conflicts.
- **Data Integrity & Security** – Databases enforce constraints and permissions, unlike file systems.

1.4. What are the responsibilities of the DBA and the database designers?

Answer:

Responsibilities of DBA and Database Designers

- **Database Administrator (DBA)**:
    1. **Database Security** – Controls user access and permissions.
    2. **Backup & Recovery** – Ensures data is regularly backed up and restorable.
    3. **Performance Optimization** – Tunes queries and indexes for efficiency.
    4. **Database Maintenance** – Monitors and updates database systems.
- **Database Designers**:
    1. **Data Modeling** – Designs database structure based on business needs.
    2. **Normalization** – Reduces redundancy and ensures efficient storage.
    3. **Defining Constraints** – Implements rules for data integrity.
    4. **Choosing Storage Techniques** – Determines indexing and partitioning strategies.

1.5. What are the different types of database end users? Discuss the main activities of each.

Answer:

Types of Database End Users and Their Activities

1. **Casual Users** – Run ad-hoc queries using GUI tools or SQL.
2. **Naive Users (Parametric Users)** – Use predefined forms and canned transactions (e.g., ATM users).
3. **Sophisticated Users** – Write complex queries and scripts (e.g., data analysts, engineers).

4. **Database Administrators (DBAs)** – Manage and maintain the database.
5. **Application Developers** – Build and optimize database-driven applications.

1.6. Discuss the capabilities that should be provided by a DBMS.

Answer:

Capabilities of a DBMS

1. **Data Storage and Retrieval** – Efficiently stores and retrieves data.
2. **Transaction Management** – Ensures ACID properties for reliability.
3. **Concurrency Control** – Manages multiple users accessing data simultaneously.
4. **Security & Authorization** – Controls user permissions and access.
5. **Backup & Recovery** – Prevents data loss and ensures system reliability.
6. **Query Processing & Optimization** – Speeds up data retrieval through indexing and execution plans.

1.7. Discuss the differences between database systems and information retrieval systems.

Answer:

Differences Between Database Systems and Information Retrieval Systems

- **Data Type**: Database systems store structured data (e.g., tables, relations), while information retrieval systems handle unstructured data (e.g., text, images, documents).
- **Query Language**: Databases use structured query languages like SQL, whereas information retrieval systems rely on keyword-based search techniques.
- **Data Integrity**: Database systems enforce strict integrity constraints and relationships, while information retrieval systems do not have rigid data consistency rules.
- **Usage**: Database systems are commonly used for transactional applications like banking and ERP systems, whereas information retrieval systems are used for document searches, search engines, and content management.
- **Precision vs. Recall**: Databases prioritize precision by retrieving exact matches, while information retrieval systems focus on recall, aiming to return as many relevant results as possible.

Exercises

1.8. Identify some informal queries and update operations that you would expect to apply to the database shown in Figure 1.2.

Answer:

Informal Queries and Update Operations

Examples of queries and updates that might apply to the database:

- **Queries:**
    1. Retrieve the names of all students enrolled in a specific course.
    2. Find all courses taught by a particular professor.
    3. List students who scored above 90% in a given subject.
- **Updates:**
    1. Enroll a new student in a course.
    2. Change a professor's assigned course.
    3. Update a student's grade for a specific subject.

1.9. What is the difference between controlled and uncontrolled redundancy? Illustrate with examples.

Answer:

Controlled vs. Uncontrolled Redundancy

- **Controlled Redundancy**:
    - Used intentionally to **improve performance** or ensure **data consistency**.
    - Example: A database may store a student's department in both the **Students** table and the **Departments** table but with proper foreign key constraints to maintain consistency.
- **Uncontrolled Redundancy**:
    - Leads to **data inconsistencies** and **wasted storage**.
    - Example: Storing a student's address in multiple tables without synchronization, leading to mismatched data

1.10. Specify all the relationships among the records of the database shown in Figure 1.2.

Answer:

Relationships Among Records in the Database

Typical relationships in an academic database:

1. **Students–Courses** (*Many-to-Many*): A student can enroll in multiple courses, and a course can have multiple students.
2. **Professors–Courses** (*One-to-Many*): A professor teaches multiple courses, but each course has only one assigned professor.
3. **Departments–Professors** (*One-to-Many*): Each professor belongs to a single department, but a department has many professors.

1.11. Give some additional views that may be needed by other user groups for the database shown in Figure 1.2.

Answer:

Additional User Views Needed

1. **Student View** – Shows only their personal information, enrolled courses, and grades.
2. **Professor View** – Displays courses they teach and students in those courses.
3. **Administrative Staff View** – Contains access to all student records, faculty assignments, and financial records.
4. **Alumni View** – Displays only personal data and degree information for past students.

1.12. Cite some examples of integrity constraints that you think can apply to the database shown in Figure 1.2.

Answer:

Examples of Integrity Constraints

1. **Primary Key Constraint** – Each student must have a unique **Student_ID**.
2. **Foreign Key Constraint** – A course's **Department_ID** must match an existing department.
3. **Not Null Constraint** – A professor must have a name and department assigned.
4. **Check Constraint** – Grades must be between 0 and 100.

1.13. Give examples of systems in which it may make sense to use traditional file processing instead of a database approach.

Answer:

When to Use Traditional File Processing Instead of a Database

1. **Small-Scale Applications** – A simple text-based inventory system for a local store.
2. **Embedded Systems** – Configuration files for embedded software in appliances.
3. **Real-Time Processing** – Logging data in high-speed trading systems where performance is critical.
4. **Single-User Applications** – Personal finance tracking tools.

a. If the name of the 'CS' (Computer Science) Department changes to 'CSSE' (Computer Science and Software Engineering) Department and the corresponding prefix for the course number also changes, identify the columns in the database that would need to be updated.

Answer:

Columns to Update When Changing 'CS' to 'CSSE'

1. **Departments Table** – Update the **Department_Name** from "CS" to "CSSE".
2. **Courses Table** – Modify the **Course_Prefix** (e.g., "CS101" → "CSSE101").
3. **Students Table** – If a department field exists, update all "CS" records to "CSSE".
4. **Professors Table** – Update the department field for all CS faculty members.

b. Can you restructure the columns in the COURSE, SECTION, and PREREQUISITE tables so that only one column will need to be updated?

Answer:

To restructure the **COURSE**, **SECTION**, and **PREREQUISITE** tables so that only one column needs to be updated when changing the department name (from "CS" to "CSSE"), follow these steps:

Step 1: Add a Department Table

Create a **Department** table that stores department information with a **Department_ID** and the corresponding **Department_Name** (e.g., "CS" and "CSSE"). This allows you to reference the department in other tables by its ID rather than by its name.

Step 2: Modify the Existing Tables

- **COURSE Table**: Replace the **Department_Name** column with a **Department_ID** that refers to the **Department** table. This way, the department is linked by ID rather than being stored as text.
- **SECTION Table**: Similarly, replace the **Department_Name** column with a **Department_ID** to indicate the department for each course section.
- **PREREQUISITE Table**: Add a **Department_ID** column here too, ensuring that the department is associated with the prerequisite courses.

Result:

When the department name changes (from "CS" to "CSSE"), you only need to update the **Department_Name** in the **Department** table. The **COURSE**, **SECTION**, and **PREREQUISITE** tables will continue to use the **Department_ID**, which remains unchanged. This structure eliminates the need to update the department in multiple places