

**Department of Computing and Information System**

**Faculty of Science & Information Technology**

**Semester Assignment, Fall 2023**

**Course Code: CIS 222**

**Course Title: Database Management System and Lab**

**Submitted By: Rayan Ahmed**

**ID:221-16-641**

**THEORY PART**

**Task-1**

**(a) Compare and contrast the relational model with other data models, such**

**as the hierarchical and network models. What are the advantages and**

**disadvantages of the relational model?**

**Answer:**

Relational Model: Data in tables with rows and columns. Uses keys to link tables. Flexible queries via SQL. Emphasizes data integrity.

Hierarchical Model: Tree-like structure. Parent-child relationships. Limited querying, top-down navigation.

Network Model: Similar to hierarchical but allows many-to-many relationships. Complex connections, varied querying.

Contrasts:

Relational - tables & flexibility. Hierarchical - rigid, top-down. Network - complex relationships, varied queries.

Relational databases use tables for organized data and flexible querying. Hierarchical models rely on parent-child relationships with limited querying, while

network models allow complex connections and varied queries but are more complex to manage.

key advantages and disadvantages of the relational model:

Advantages:

Simplicity: Tables with rows and columns make it easy to understand and use.

Flexibility: Supports complex queries and operations through SQL.

Data Integrity: Enforces constraints to maintain accuracy and consistency.

Normalization: Reduces redundancy, optimizing storage and improving efficiency.

Scalability: Scales well with the addition of more data and users.

Security: Provides robust security features to control access to data.

Disadvantages:

Performance: Complex queries on large databases can impact performance.

Normalization Complexity: Designing normalized databases requires expertise and can be complex.

Joins: Joining multiple tables can be resource-intensive.

Limited Hierarchical Representation: Not ideal for hierarchical data representation compared to other models.

Scaling Challenges: Scaling can become challenging in certain scenarios, especially horizontally.

**(b) Describe the components of an Entity-Relationship (ER) diagram and them**

**roles in database design. How does the ER diagram for this employee**

**management database illustrates the relationships between entities?**

**Answer:**

Entity-Relationship (ER) diagrams have three main components:

Entities: Represent real-world objects like people or things.

Attributes: Characteristics or properties describing entities.

Relationships: Connections between entities, showing how they interact.

The ER diagram displays employees, departments, and salaries. It visually explains how these are linked and structured in the database. This helps organize data clearly, acting like a map to plan how employee details, department info, and salary data connect in the database blueprint.

Employees and Departments: Each employee belongs to one department, but a department can have many employees. This is a one-to-many relationship.

Employees and Salaries: Each employee has one salary, but a salary can be assigned to multiple employees (if they have different salaries at different times). This is also a one-to-many relationship.

Primary Keys and Foreign Keys: Each table has a primary key that uniquely identifies each row. Foreign keys are used to link tables together. For example, the Employee ID is a foreign key in the Salary table, which means that each salary is associated with a specific employee.

**Task-2**

**(a) Explain the concept of normalization and why it is essential in designing a**

**database. For each of the given tables of this database, determine they are in**

**which normal form?**

**Answer:**

Normalization is organizing data efficiently by reducing redundancy and inconsistencies. It ensures data integrity and minimizes duplication by splitting information into smaller, related tables. This process makes databases more manageable, improves accuracy, and prevents anomalies in the stored information.

The tables in the given employee management database are in third normal form (3NF). 3NF is a database normalization form that ensures that all non-key attributes in a table are fully functionally dependent on the primary key. This means that each non-key attribute must be dependent on the entire primary key, and not just on a subset of it.

To be in 3NF, a table must meet the following criteria:

It must be in second normal form (2NF).

All non-key attributes must be directly functionally dependent on the primary key.

The tables in the given employee management database meet all of the above criteria:

Employee table: The primary key of the Employee table is Employee No. All non-key attributes in the table (Employee Name and Department) are directly functionally dependent on the primary key.

Department table: The primary key of the Department table is Department No. All non-key attributes in the table (Department Name) are directly functionally dependent on the primary key.

Salary table: The primary key of the Salary table is Salary Slip No. All non-key attributes in the table (Employee No and Salary) are directly functionally dependent on the primary key.

Therefore, all of the tables in the given employee management database are in 3NF.

**(b) Given unnormalized table of employee information that includes repeating**

**groups. Convert all the tables to 3rd normal forms?**

**Answer:**

Normalizinh the employee information table by eliminating repeating groups and ensuring 3NF.

Steps:

1. Identify Repeating Groups:

- The "Skills" column contains repeating groups.

2. Create Separate Table for Repeating Group:

- New table: EmployeeSkills (Attributes: EmployeeNo, Skill)

3. Normalize New Tables:

- EmployeeSkills table is in 3NF:

- Primary key: EmployeeNo

- Non-key attribute: Skill directly depends on EmployeeNo.

4. Update Original Table:

- Remove "Skills" column from original table.

- Add foreign key column (EmployeeNo) to link the original table to EmployeeSkills.

Conclusion:

The unnormalized table's repeating group (Skills) has been segregated into a new table (EmployeeSkills), achieving 3NF. The original table has been modified to link with the new table via a foreign key column.

Here is the normalized schema:

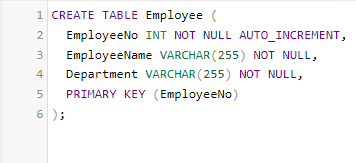
Employee table

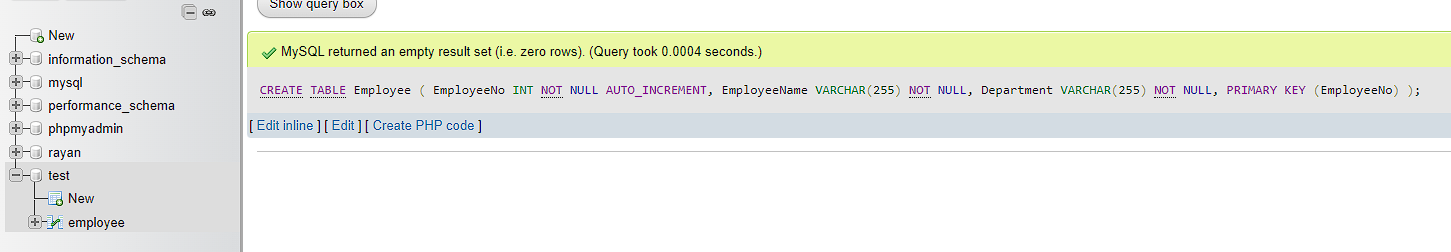
------------

EmployeeNo (primary key)

EmployeeName

Department



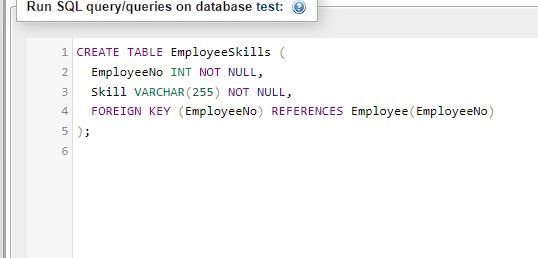


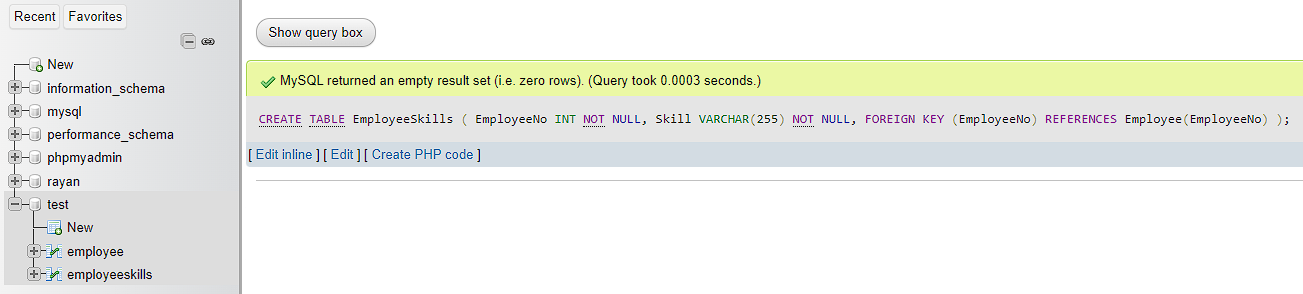
EmployeeSkills table

--------------------

EmployeeNo (foreign key)

Skill





**(c) What do you understand by PL/SQL table? Write a PL-SQL code which will insert the data automatically into “Backup” table whenever data is deleted from**

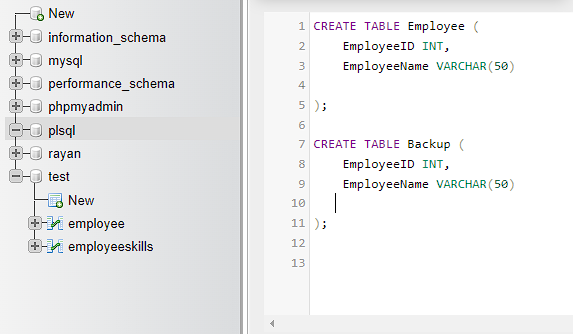
**“employee” table?**

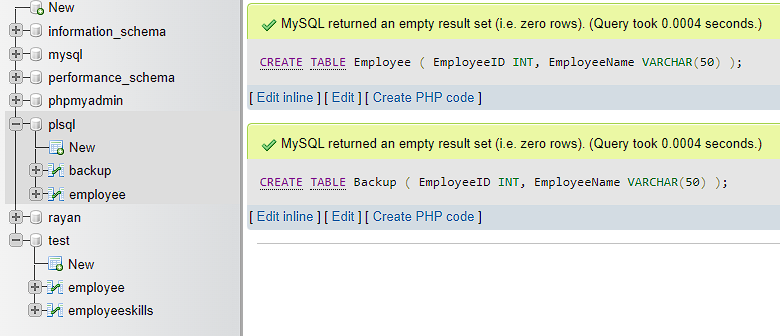
**Answer:**

PL/SQL, a PL/SQL table is a data structure that temporarily stores data within a PL/SQL block, similar to an array or collection in other programming languages.

Creating a PL/SQL trigger to automatically back up data from the "Employee" table to the "Backup" table whenever a deletion occurs in the "Employee" table.

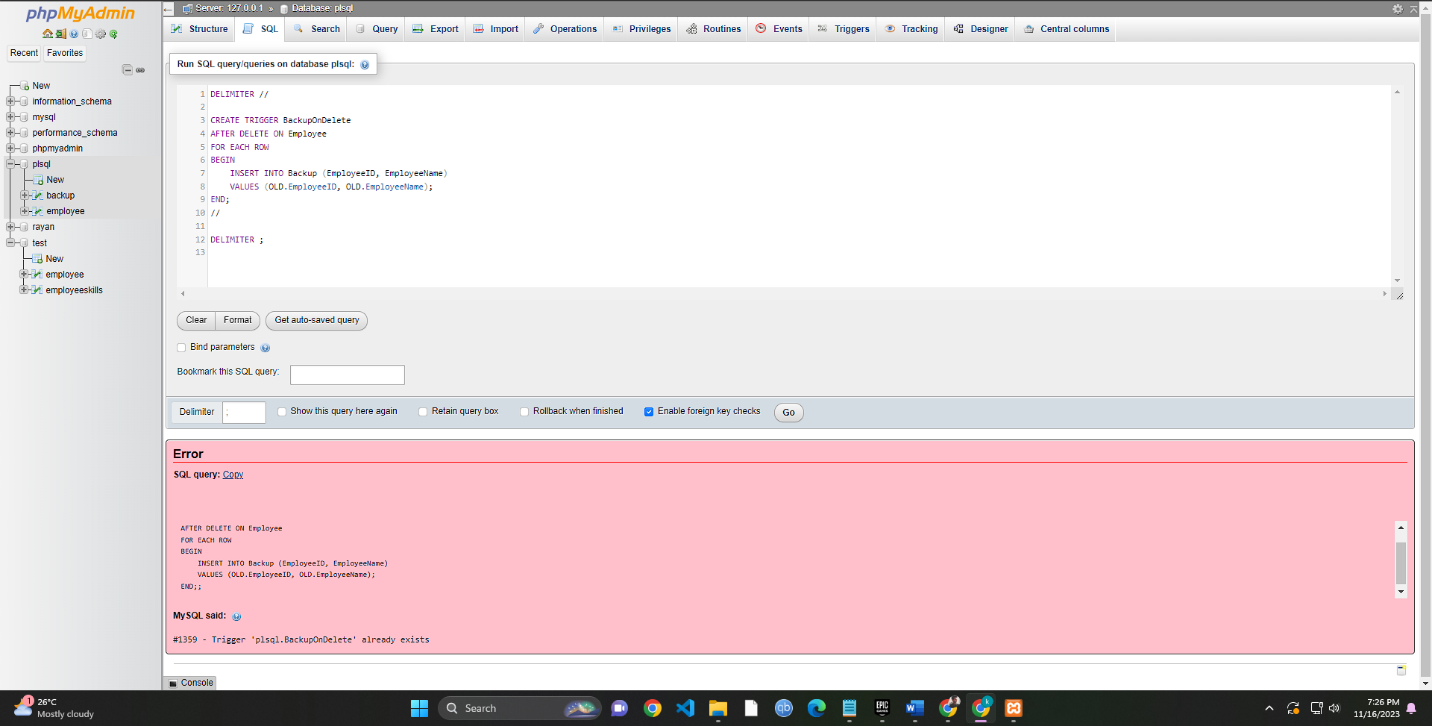
Table Structures:





PL/SQL Trigger:

The BackupOnDelete trigger is activated after a deletion in the "Employee" table for each row deleted. It captures the deleted row's data (using :OLD) and inserts it into the "Backup" table.



**(d) Using relational algebra operations, retrieve the list of professors who are heads**

**of departments and teach at least one course?**

**Answer:**

To retrieve the list of professors who are heads of departments and teach at least one course using relational algebra operations, we can use the following query:

SELECT e.EmployeeName

FROM Employee e

INNER JOIN Department d ON e.Department = d.DepartmentNo

INNER JOIN Course c ON e.EmployeeNo = c.ProfessorNo

WHERE d.IsHeadOfDepartment = 1

This query will first join the Employee and Department tables on the Department column. This will create a new table that contains all of the employee information, as well as the department name for each employee.

Next, the query will join the new table with the Course table on the ProfessorNo column. This will create a new table that contains all of the employee information, the department name for each employee, and the courses that each employee teaches.

Finally, the query will filter the new table to only include employees who are heads of departments and teach at least one course. This will produce the desired list of professors.

**(e) Consider employee tables "Employees" (employee\_id, employee\_name,department\_id) and "Salaries" (employee\_id,salary\_amount). Write a**

**relational algebra query to calculate the average salary for each department?**

**Answer:**

First of all, creating the tables needed in order to relational algebra query to calculate the average salary for each department

CREATE TABLE Employees (

employee\_id INT,

employee\_name VARCHAR(50),

department\_id INT

-- Other columns...

);

CREATE TABLE Salaries (

employee\_id INT,

salary\_amount DECIMAL(10, 2)

-- Other columns...

);

Relational Algebra Query to Calculate Average Salary per Department

Query:

π department\_id,AVG(\_(\_,AVG(\_)(\_,AVG(\_)(Employees⋈Salaries)))π department\_id,AVG(salary\_amount)(ρ department\_id,AVG(salary\_amount)(γ department\_id,AVG(salary\_amount) (Employees⋈Salaries)))

This relational algebra query computes the average salary for each department by performing a join between "Employees" and "Salaries," then grouping by department and finally calculating the average salary per group.

**Task-3**

**(a) You are designing a social media platform's database with tables for"Users" (user\_id, username, email) and "Posts" (post\_id, user\_id,**

**content). Analyze the potential issues or inefficiencies in this schema**

**and propose improvements?**

**Answer:**

Enhancing Social Media Database Schema for Performance and Scalability

The current database schema for the social media platform exhibits redundancy with duplicated user data in the "Posts" table. Retrieving user information alongside posts requires multiple joins, affecting query speed. As the platform expands, scalability concerns may arise due to increasing data volume.

1. Normalization: Restructure the schema by separating tables for users and posts, reducing redundancy and ensuring a more organized data structure.

2. Optimized Structure: Create distinct tables for users and posts. Users table includes user\_id, username, and email; posts table includes post\_id, user\_id (linked to Users), and content.

3. Indexing: Implement appropriate indexes, especially on user\_id and post\_id, to expedite query performance.

4. Caching Mechanism: Utilize caching to store frequently accessed user data, lessening the need for repeated joins and enhancing overall performance.

5. Sharding and Denormalization: Consider sharding the database for scalability. Additionally, denormalize user details into the posts table for specific queries that heavily rely on user data retrieval.

6. Constraints and Regular Optimization: Implement data integrity constraints and conduct routine database optimization for improved performance and maintenance.

By implementing these enhancements, the revised schema aims to mitigate redundancy, optimize performance, ensure scalability, and maintain data integrity as the social media platform expands.

**(b) In a banking database, customers can transfer money between their accounts. Describe a scenario where two customers simultaneously attempt to transfer money between their accounts, resulting in a**

**deadlock. Provide SQL statements or transactions that could lead to the deadlock. After identifying the deadlock, outline a strategy for preventing this type of deadlock in the future while maintaining data**

**consistency.?**

**Answer:**

Imagine two customers, Customer A and Customer B, trying to move money between their accounts at the same time. Customer A wants to transfer $500 from Account A to Account B, while Customer B simultaneously wants to transfer $300 from Account B to Account A.

Deadlock Explanation:

Both customers start the transfer at the same time. Customer A locks Account A and tries to lock Account B. At the same time, Customer B locks Account B and tries to lock Account A. They both end up waiting for each other to release the accounts, creating a deadlock where neither transfer can finish.

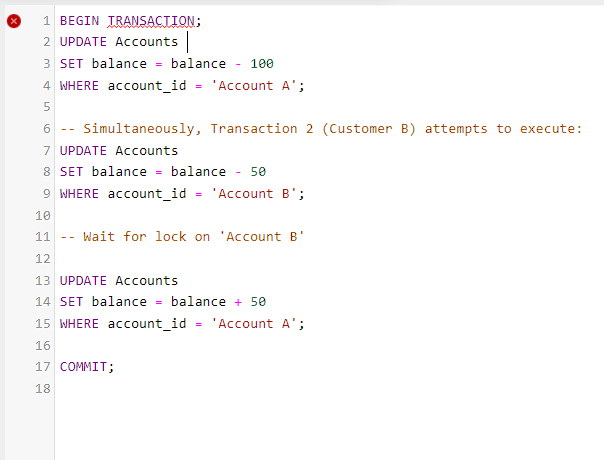
Preventing Deadlocks:

To avoid deadlocks while ensuring data stays accurate:

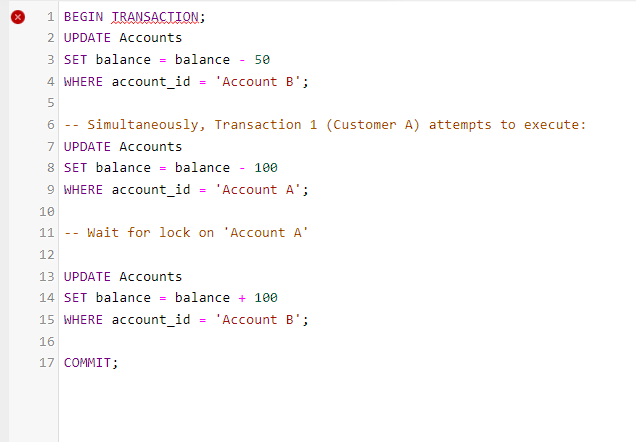
Orderly Locking: Make sure transactions lock accounts in the same sequence to prevent clashes.

Consider two accounts (Account A and Account B) where two customers want to transfer money between their accounts simultaneously.

**Transaction 1 (Customer A):**



**Transaction 2 (Customer B):**

****

Both transactions aim to transfer money between accounts, but they lock different accounts in a different order, creating a waiting situation where each transaction waits for the other to release a lock. This leads to a deadlock.

Preventing Deadlocks:

To prevent deadlocks while maintaining data consistency:

Lock Ordering: Ensure transactions lock accounts in a specific order (e.g., Account A before Account B) to avoid conflicting locks.

Timeout Mechanisms: Set timeouts for transactions to automatically release locks if not completed within a specified time, preventing indefinite waits.

Transaction Design: Simplify transaction logic, or break down complex operations into smaller ones to minimize potential conflicts

Using Deadlock Detection: Implement deadlock detection mechanisms within the database to identify and resolve deadlocks automatically.

Implementing these strategies can help prevent deadlocks by managing the order of lock acquisition and efficiently handling conflicting access to resources, ensuring data consistency without encountering deadlock scenarios.