**Assignment and practice II:**

**Submission date: (8days)next Friday from today July3rd.**

**Question1:**  
Create a table using the below details and set id as primary key for employee table.  
employee(id, name, address,salary,company\_name,job\_title)  
Insert 5 rows on it.  
1. Display all employee names and salary whose salary is greater than minimum salary of  
the company and job title starts with 'M‘.  
2. Write a query to find all the employees who work in the same job\_title as Ram.

CREATE DATABASE company;

USE company;

CREATE TABLE employee (

id INT PRIMARY KEY,

ename VARCHAR(30),

address VARCHAR(50),

salary FLOAT,

company\_name VARCHAR(30),

job\_title VARCHAR(30)

);

INSERT INTO employee VALUES

(1, 'Ram', 'Kathmandu', 30000, 'ABC Corp', 'Manager'),

(2, 'Shyam', 'Lalitpur', 25000, 'XYZ Ltd', 'Marketing Executive'),

(3, 'Hari', 'Bhaktapur', 28000, 'ABC Corp', 'Mechanic'),

(4, 'Sita', 'Pokhara', 32000, 'XYZ Ltd', 'Manager'),

(5, 'Gita', 'Biratnagar', 22000, 'ABC Corp', 'Mason');

SELECT ename, salary FROM employee AS e WHERE salary >

(SELECT MIN(salary) FROM employee WHERE company\_name = e.company\_name)

AND job\_title LIKE 'M%';

SELECT \* FROM employee WHERE job\_title =

(SELECT job\_title FROM employee WHERE ename = 'Ram');  
  
**Question2:**  
Create a table using the below details and set id as primary key for student table.  
student(id, name, course, subject\_id, mark)  
subject(subject\_id, subject)  
Insert 5 rows on both tables.  
1. Select the student details along with subject and mark.  
2. Group the students based on the course.

CREATE DATABASE school;

USE school;

CREATE TABLE ssubject (

subject\_id INT PRIMARY KEY,

ssubject VARCHAR(30)

);

CREATE TABLE student (

id INT PRIMARY KEY,

sname VARCHAR(30),

course VARCHAR(30),

subject\_id INT,

mark INT,

FOREIGN KEY (subject\_id) REFERENCES ssubject(subject\_id)

);

INSERT INTO ssubject VALUES

(1, 'Math'),

(2, 'Science'),

(3, 'English'),

(4, 'Computer'),

(5, 'Nepali');

INSERT INTO student VALUES

(101, 'Amit', 'BSc', 1, 85),

(102, 'Bina', 'BSc', 2, 75),

(103, 'Chetan', 'BBA', 3, 65),

(104, 'Diya', 'BBA', 4, 80),

(105, 'Elina', 'BSc', 5, 90);

SELECT s.id, s.sname, s.course, sub.ssubject, s.mark

FROM student AS s

JOIN ssubject AS sub

ON s.subject\_id = sub.subject\_id;

SELECT course, COUNT(\*) AS total\_students

FROM student

GROUP BY course;  
  
**Question3:**  
Create a table using the below details and set id as primary key for faculty table.  
faculty(id, name address, subject, salary)  
Insert 5 rows on it.  
1. Find total rows, sum, average, maximum, minimum salary of faculty.  
2. Write a query to display all the faculty and salary whose salary is greater than average salary of all faculty.

CREATE DATABASE university;

USE university;

CREATE TABLE faculty (

id INT PRIMARY KEY,

fname VARCHAR(30),

address VARCHAR(50),

fsubject VARCHAR(30),

salary FLOAT

);

INSERT INTO faculty VALUES

(1, 'Arjun', 'Kathmandu', 'Math', 30000),

(2, 'Bijay', 'Lalitpur', 'Science', 25000),

(3, 'Chetana', 'Pokhara', 'English', 32000),

(4, 'Dipak', 'Bhaktapur', 'Computer', 28000),

(5, 'Elisha', 'Chitwan', 'Nepali', 27000);

SELECT

COUNT(\*) AS total\_rows,

SUM(salary) AS total\_salary,

AVG(salary) AS avg\_salary,

MAX(salary) AS max\_salary,

MIN(salary) AS min\_salary

FROM faculty;

SELECT fname, salary FROM faculty WHERE salary >

(SELECT AVG(salary) FROM faculty);  
  
**Question4:**  
Create a table using the below details set id as primary key for teacher table.  
teacher(id, name, address,salary)  
subject(subject\_id, subject\_name)  
Insert 5 rows on it.  
1. Add a new column subject\_id on the teacher table.  
2. Create a view from the teacher using the name and subject\_name column.

CREATE DATABASE college;

USE college;

CREATE TABLE ssubject (

subject\_id INT PRIMARY KEY,

subject\_name VARCHAR(30)

);

CREATE TABLE teacher (

id INT PRIMARY KEY,

tname VARCHAR(30),

address VARCHAR(50),

salary FLOAT

);

INSERT INTO ssubject VALUES

(1, 'Math'), (2, 'Science'), (3, 'English'), (4, 'Computer'), (5, 'Nepali');

INSERT INTO teacher VALUES

(1, 'Anil', 'Kathmandu', 25000),

(2, 'Bimala', 'Lalitpur', 27000),

(3, 'Chiran', 'Pokhara', 23000),

(4, 'Dikshya', 'Bhaktapur', 26000),

(5, 'Erika', 'Chitwan', 28000);

ALTER TABLE teacher ADD subject\_id INT;

ALTER TABLE teacher ADD FOREIGN KEY (subject\_id) REFERENCES ssubject(subject\_id);

UPDATE teacher SET subject\_id = 1 WHERE id = 1;

UPDATE teacher SET subject\_id = 2 WHERE id = 2;

UPDATE teacher SET subject\_id = 3 WHERE id = 3;

UPDATE teacher SET subject\_id = 4 WHERE id = 4;

UPDATE teacher SET subject\_id = 5 WHERE id = 5;

CREATE VIEW teacher\_subject\_view AS

SELECT t.tname, s.subject\_name

FROM teacher AS t

JOIN ssubject AS s ON t.subject\_id = s.subject\_id;

SELECT \* FROM teacher\_subject\_view;

**Question5:**

What is tuple realtion calculus ?  Given the following schema, write tuple relational calculus for selecting name and address of employee who are working in a company having Cid=E01 ,  
Employee(Eid, Ename, Address, Cid)   
Company(Cid, CName)

**Question6:**

What is relational database? Explain different characteristic of a relation. Defain domain constraint.

1. Relational Database

The **Relational Model** represents the database as a **collection of relations**. A relation, in simple terms, is a table of values. The model uses tables and relationships among those tables. Each row in the table represents a collection of related data values. The most popular data model in use today is the relational data model.

2. Characteristics of a Relation

A relation (or table) must adhere to several characteristics to maintain integrity and structure in the relational model:

1. **Distinct Name:** Each relation in the database must possess a **distinct or unique name** that differentiates it from other relations.

2. **Unique Attribute Names:** A relation cannot have two attributes (columns) with the same name.

3. **Atomic Values:** Each tuple (row) must have **exactly one data value for an attribute**.

4. **No Duplicate Tuples:** Duplicate tuples must not be present in the relation.

5. **Order Insensitivity:** Tuples in a relation do not follow a significant order, as the relation is not order-sensitive. Similarly, the attributes of a relation do not have to follow a significant order.

3. Domain Constraint Definition

**Domain constraints** are integrity rules that specify that the value of each attribute must be an **atomic value** drawn from the attribute's defined domain. The domain is defined as a set of acceptable values that a column is allowed to contain, based on the required data type for that column.

• **Detail:** Atomic value means that each value in the domain is indivisible as far as the relational model is concerned.

• **Example:** The domain for Marital Status might restrict values to only {'Married', 'Single', 'Divorced'}

**Question7:**

What are the characteristics of DBMS? Explain.

A **Database Management System (DBMS)** is a software system designed for efficient data management, providing several key advantages and characteristics over traditional file systems.

The main characteristics and advantages of using a DBMS include:

1. **Data Redundancy Control:** The DBMS approach minimizes the duplication of data (redundancy), often through processes like normalization. File systems, conversely, often contain multiple copies of the same data.

2. **Data Consistency and Integrity:** The DBMS ensures that all data adheres to defined rules and constraints, automatically maintaining **data consistency**. It provides mechanisms (constraints and validation rules) to uphold the accuracy and **integrity** of the data.

3. **Data Sharing and Concurrency Control:** A DBMS supports access by **multiple users/applications simultaneously** (data sharing). It employs **concurrency control** mechanisms to manage these simultaneous operations, preventing conflicts and ensuring consistent updates.

4. **Security Features:** The system offers robust **security features**, including user authentication, authorization, and various levels of access control, ensuring that only authorized users can interact with the data.

5. **Backup and Recovery:** A DBMS has built-in mechanisms and a recovery subsystem responsible for reliably backing up data and restoring the database to a consistent state following any hardware or software failure.

6. **Data Independence:** The DBMS separates the description of data (schema) from how the data is physically stored (physical data independence) and logically organized (logical data independence). This allows changes in storage methods or logical design without affecting existing application programs.

7. **Self-Describing Nature:** The database system itself contains not only the database but also a complete description of the structure, schema, storage format, and data types (called metadata), which is stored in the DBMS catalog.

8. **Support for Multiple Views:** A DBMS provides facilities for defining multiple views, allowing different types of users to have diverse, customized perspectives or subsets of the data.

**Question8:**

What is difference between Entities and Entity sets? Explain with example.

The distinction between an Entity and an Entity Set is based on scope and quantity.

1. Entity

An **Entity** is defined as a **thing or object** in the real world with an independent existence. An entity may be an object with a physical existence (like an employee) or one with a conceptual existence (like a job or a course).

• **Detail:** A particular entity will have a value for each of its attributes (properties).

• **Example:** A single specific employee, such as "Employee John Doe with SSN 1234," is an entity.

2. Entity Set (or Entity Type)

An **Entity Type** (or Entity Set) defines a **collection (or set) of entities** that share the same attributes. An Entity Set is often represented using the same name as the Entity Type, even though they represent separate concepts.

• **Entity Type:** This defines the *structure* of the collection. For example, the **EMPLOYEE** type defines the structure (Name, Age, Salary) for all employees.

• **Entity Set:** This is the actual **collection of entities** of a particular type present in the database at any given point in time.

• **Example:** The **EMPLOYEE entity set** refers to the current collection of *all employee entities* stored in the database.

**In summary:** If **Employee** refers to the specific individual (John Doe), the **EMPLOYEE set** refers to the logical *table structure* (the definition) and the *set of all records* (the contents) currently holding all employees.

**Question9:**

What is data abstraction? What are different levels of data abstraction? Brief it.

**Data abstraction** refers to a **hiding mechanism** that suppresses the low-level details regarding data organization and storage. Its purpose is to highlight only the **essential features** of the data to improve understanding.

• Data abstraction is a **fundamental characteristic of the database approach**.

• It is typically achieved through the **three-schema architecture**, the goal of which is to separate user applications from the physical database structure and enable data independence.

2. Different Levels of Data Abstraction

The three-schema architecture defines three distinct levels of abstraction:

1. **Internal Level (Physical Schema):**

    ◦ **Level:** Lowest level.

    ◦ **Description:** Describes **how the data is physically stored** in the database.

    ◦ **Content:** Includes full details of storage, access paths, file structures, and indexing.

    ◦ **Example:** Student data is stored as binary records in files on a hard disk, sorted by roll number.

2. **Conceptual Level (Logical Schema):**

    ◦ **Level:** Middle level.

    ◦ **Description:** Describes the **structure of the entire database** for the community of users. It hides the physical storage details.

    ◦ **Content:** Defines all **entities, relationships, data types, and integrity constraints** (like primary and foreign keys).

    ◦ **Example:** The definition of the logical table Student(Roll, Name, Major) with Roll as the primary key.

3. **External Level (View Schema):**

    ◦ **Level:** Highest level.

    ◦ **Description:** Describes **how individual users or applications see the data**. Each user can have a customized view.

    ◦ **Content:** Defines specific subsets of the database tailored to a particular user group, hiding the rest of the database structure for simplification and security.

    ◦ **Example:** A teacher might only see Student(RollNo, Name), while the finance department sees Student(RollNo, FeeStatus).

**Question 10:**

Explain constraints and characteristics of specialization and generalization of data model.

Specialization and Generalization are concepts used in the Enhanced ER (EER) model to handle subclasses and superclasses.

1. Characteristics (Definitions)

• **Specialization:** This is a **top-down approach**. A higher-level entity (superclass) is broken down into two or more lower-level entities (subclasses). It aims to maximize the differences between attributes of the members.

    ◦ **Example:** Breaking down the entity Employee into subclasses Developer and Tester based on their role.

• **Generalization:** This is a **bottom-up approach**. Two or more lower-level entities (subclasses) are combined to form a higher-level entity (superclass).

    ◦ **Example:** Combining entities Tiger, Lion, and Elephant to form the generalized entity Animal.

2. Constraints on Specialization and Generalization

There are three main categories of constraints that apply to these concepts:

A. Membership Constraints

These constraints determine how an entity becomes a member of a subclass:

1. **Condition-Defined Constraints:** Membership in a subclass is determined automatically by evaluating a condition or predicate applied to the attributes of the superclass.

2. **User-Defined Constraints:** Membership in a subclass is explicitly specified for each entity in the superclass by the database users during operations (like adding a tuple to a subclass).

B. Disjoint Constraints

These constraints specify whether an entity can belong to more than one subclass within the specialization:

1. **Disjoint Constraint (d):** Specifies that an entity can be a member of **at most one subclass** of the specialization/generalization.

2. **Overlapping Constraint (o):** Specifies that the subclasses are **not constrained to be disjoint**, meaning the same entity may belong to **more than one subclass** simultaneously.

C. Completeness Constraints

These constraints specify whether every entity in the superclass must be a member of some subclass:

1. **Total Participation Constraint:** Specifies that **every entity in the superclass must be a member of at least one subclass** in the specialization/generalization. (Represented by a double line in the EER diagram.)

2. **Partial Participation Constraint:** Allows an entity in the superclass to **not belong to any subclass** in the specialization/generalization. (Represented by a single line.)

**Question 11:**

Explain the difference between “Join” and “Natural Join”, of algebriac operations with example.

A screenshot of a computer program

AI-generated content may be incorrect.

**Question 12:**

Explain the difference between 3 different Outer joins of algebriac operations with example.

Outer Joins are extensions of the standard join operation designed to keep tuples that do not satisfy the matching criteria, thereby dealing with missing information.

1. Left Outer Join ()

• Definition: This operation ensures that all tuples in the left relation are preserved in the result, regardless of whether a matching tuple is found in the right relation.

• Missing Data Handling: If there is no matching tuple in the right relation, the attributes corresponding to the right relation in the join result are filled with NULL values.

• Example (Conceptual): If you Left Outer Join Department (Left) and Staff (Right), and Department 3 ("Economics") has no staff member yet, the output will list Department 3, and all the staff-related columns will show NULL.

2. Right Outer Join ()

• Definition: This operation ensures that all tuples in the right relation are kept in the result.

• Missing Data Handling: If there is no matching tuple found in the left relation for a tuple in the right relation, the attributes corresponding to the left relation are filled with NULL values.

• Example (Conceptual): If you Right Outer Join Department (Left) and Staff (Right), and a staff member exists with a Dept.id that is not currently in the Department table, that staff member's information will be included, and all Department fields will show NULL.

3. Full Outer Join ()

• Definition: This operation includes all tuples from both the left-hand relation and the right-hand relation. It includes all matching tuples plus all non-matching tuples from both sides.

• Missing Data Handling: Non-matching attributes from either side are filled with NULL values.

• Example (Conceptual): If a department exists with no staff (Left non-match) AND a staff member exists attached to a non-existent department (Right non-match), the Full Outer Join includes both of these scenarios, filling the opposite table's columns with NULLs in each case.