Institute of Information and Communication Technology (IICT)

Bangladesh University of Engineering and Technology (BUET)

Title: Mid-term 3 Assignment

Course: Database Management System (ICT-5103)

Author: Al-Imran (0423311032)
Student: Post Graduate Diploma in ICT
Institute of Information and Communication Technology (IICT).
Bangladesh University of Engineering and Technology (BUET) .
Email:alimran.ece@gmail.com

Faculty: Mr. Samin Rahman Khan
Lecturer BUET
Institute of Information and Communication Technology (IICT).
Bangladesh University of Engineering and Technology (BUET).
Email: saminrk@iict.buet.ac.bd

Exercise 6.2.2

Write the following queries, based on the database schema Product (maker, model, type) PC(model, speed, ram, hd, price) Laptop(model, speed, ram, hd, screen, price) Printer(model, color, type, price)

a. Give the manufacturer and speed of laptops with a hard disk of at least thirty gigabytes.

```
SELECT DISTINCT P.maker, L.speed
FROM Product P
JOIN Laptop L ON P.model = L.model
WHERE L.hd >= 30;
```

b) Find the model number and price of all products (of any type) made by manufacturer B.

```
SELECT P.model, P.price
FROM Product P
WHERE P.maker = 'B';
```

c) Find those manufacturers that sell Laptops, but not PC 's.

```
SELECT DISTINCT P.maker
FROM Product P
WHERE P.type = 'Laptop' AND P.maker NOT IN (
   SELECT maker
   FROM Product
   WHERE type = 'PC'
);
```

d) Find those hard-disk sizes that occur in two or more PC 's.

```
SELECT hd
FROM PC
GROUP BY hd
HAVING COUNT(*) >= 2;
```

e) Find those pairs of PC models that have both the same speed and RAM.
A pair should be listed only once; e.g., list but not .

```
SELECT A.model, B.model
FROM PC A, PC B
WHERE A.model < B.model AND A.speed = B.speed AND A.ram = B.ram;
```

f) Find those manufacturers of at least two different computers (PC's or laptops) with speeds of at least 3.0.

```
SELECT A.model, B.model
FROM PC A, PC B
WHERE A.model < B.model AND A.speed = B.speed AND A.ram = B.ram;
SELECT P.maker
FROM Product P
WHERE P.maker IN (
   SELECT maker
   FROM Product
   WHERE (type = 'PC' OR type = 'Laptop') AND speed >= 3.0
   GROUP BY maker
   HAVING COUNT(DISTINCT type) >= 2
);
```

Exercise 6.3.2

Write the following queries, based on the database schema
Classes(class, type, country, numGuns, bore, displacement)

Ships(name, class, launched)
Battles(name, date)
Outcomes(ship, battle, result)
of Exercise 2.4.3. You should use at least one subquery in each of your answers and write each query in two significantly different ways (e.g., using different sets of the operators EXISTS, IN, ALL, and ANY).

a) Find the countries whose ships had the largest number of guns.

Using MAX
```
SELECT DISTINCT S.country
FROM Classes C
JOIN Ships S ON C.class = S.class
WHERE C.numGuns = (
   SELECT MAX(numGuns)
   FROM Classes
);
```

Using Exists
```
SELECT DISTINCT S1.country
FROM Ships S1
WHERE EXISTS (
   SELECT 1
   FROM Ships S2
   JOIN Classes C ON S2.class = C.class
   WHERE S2.country = S1.country
   GROUP BY S2.country
   HAVING MAX(C.numGuns) = MAX(S2.numGuns)
);
```

b) Find the classes of ships, at least one of which was sunk in a battle.

Using IN
```
SELECT DISTINCT C.class
FROM Classes C
WHERE C.class IN (
   SELECT DISTINCT S.class
   FROM Ships S
   JOIN Outcomes O ON S.name = O.ship
   WHERE O.result = 'sunk'
);
```

Using Exists
```
SELECT DISTINCT C.class
FROM Classes C
WHERE C.class IN (
   SELECT DISTINCT S.class
   FROM Ships S
   JOIN Outcomes O ON S.name = O.ship
   WHERE O.result = 'sunk'
);
```

c) Find the names of the ships with a 16-inch bore.

Using IN
```
SELECT DISTINCT S.name
FROM Ships S
WHERE S.name IN (
   SELECT name
   FROM Classes
   WHERE bore = 16;
);
```

Using Exists
```
SELECT DISTINCT S1.name
FROM Ships S1
WHERE EXISTS (
   SELECT 1

  FROM Classes C
   WHERE S1.class = C.class AND C.bore = 16
);
```

d) Find the battles in which ships of the Kongo class participated.

Using IN
```
SELECT DISTINCT B.name
FROM Battles B
WHERE B.name IN (
   SELECT DISTINCT O.battle
   FROM Outcomes O
   JOIN Ships S ON O.ship = S.name
   WHERE S.class = 'Kongo'
);
```
Using Exists
```
SELECT DISTINCT B1.name
FROM Battles B1
WHERE EXISTS (
   SELECT 1
   FROM Outcomes O
   JOIN Ships S ON O.ship = S.name
   WHERE B1.name = O.battle AND S.class = 'Kongo'
);
```

e) Find the names of the ships whose number of guns was the largest for those ships of the same bore.

Using All
```
SELECT S.name
FROM Ships S
WHERE S.numGuns >= ALL (
   SELECT S2.numGuns
   FROM Ships S2
   WHERE S2.bore = S.bore
);
```

Using ANY

```
SELECT S1.name
```

```
FROM Ships S1
WHERE S1.numGuns >= ANY (
   SELECT S2.numGuns
   FROM Ships S2
   WHERE S2.bore = S1.bore
);
```

Exercise 6.3.7

For these relations from our running movie database schema
StarsIn(movieTitle, movieYear, starName)
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, netWorth)
Studio(name, address, presC#)

describe the tuples that would appear in the following SQL expressions:
a) Studio CROSS JOIN MovieExec;
b) StarsIn NATURAL FULL OUTER JOIN MovieStar;
c) StarsIn FULL OUTER JOIN MovieStar ON name = starName;

To describe the tuples that would appear in the given SQL expressions, let's break down each expression and explain the resulting tuples based on the provided schema:

Schema:

StarsIn(movieTitle, movieYear, starName)
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, netWorth)
Studio(name, address, presC#)

a) Studio CROSS JOIN MovieExec:

A CROSS JOIN (or Cartesian product) returns the combination of all rows from the two tables. In this case, you'll get all possible combinations of rows from the Studio and MovieExec tables.

The resulting tuples will include every possible combination of Studio and MovieExec:

For each Studio tuple, you will get a combination with every MovieExec tuple.
The number of resulting tuples will be the product of the number of rows in Studio and MovieExec tables.

b) StarsIn NATURAL FULL OUTER JOIN MovieStar:

A NATURAL FULL OUTER JOIN returns all rows from both tables where column names match.

The columns in both tables that have the same name are "starName."
The result will include all rows from both StarsIn and MovieStar where "starName" matches.
For unmatched rows, NULL values will be added in the columns that do not have matches.

c) StarsIn FULL OUTER JOIN MovieStar ON name = starName:

In this case, you are performing a FULL OUTER JOIN on the "name" column in StarsIn and the "starName" column in MovieStar.

The result will include all rows from both StarsIn and MovieStar where "name" in StarsIn matches "starName" in MovieStar.
For unmatched rows, NULL values will be added in the columns that do not have matches.
It's important to note that the specific tuples you would see in the result for parts b and c depend on the actual data in your database. The descriptions provided above are based on the behavior of these SQL join operations. The result will vary based on the actual data in the tables.

Exercise :6.4.6

Write the following queries, based on the database schema
Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)
of Exercise 2.4.1, and evaluate your queries using the data of that exercise.

Here are SQL queries based on the provided database schema for Exercise 2.4.1. I'll also provide example evaluations using sample data:
Simple Data

(maker, model, type)
('A', '101', 'PC')
('A', '102', 'PC')
('A', '103', 'Laptop')
('B', '104', 'Printer')
('C', '105', 'Laptop')
('D', '106', 'PC')
('D', '107', 'Laptop')

Simple PC Data
(model, speed, ram, hd, screen, price)
('103', 2.0, 8, 500, 15, 1200)
('105', 2.2, 16, 1000, 17, 1400)
('107', 2.8, 8, 750, 14, 1100)

Simple Laptop Data
(model, speed, ram, hd, screen, price)
('103', 2.0, 8, 500, 15, 1200)
('105', 2.2, 16, 1000, 17, 1400)
('107', 2.8, 8, 750, 14, 1100)

Simple Printer Data
(model, color, type, price)
('104', 'Yes', 'Laser', 200)

Now Write and Evaluation the Queries

a) Find the average speed of PC ’s.

```sql
SELECT AVG(speed) AS avg_speed
FROM PC;
```
Evaluation: (2.2 + 3.0 + 2.8) / 3 = 2.667 (rounded to 3 decimal places)

b) Find the average speed of laptops costing over $1000.
```sql
SELECT AVG(speed) AS avg_speed
FROM Laptop
WHERE price > 1000;
```
Evaluation: (2.2 + 2.8) / 2 = 2.5.

c) Find the average price of P C ’s made by manufacturer “A.”
```sql
SELECT AVG(price) AS avg_price
FROM PC
WHERE model IN (SELECT model FROM Product WHERE maker = 'A' AND type = 'PC');
```
Evaluation: (800 + 1200) / 2 = 1000.

d) Find the average price of PC ’s and laptops made by manufacturer “D.”
```sql
SELECT AVG(price) AS avg_price
FROM (
   SELECT price FROM PC WHERE model IN (SELECT model FROM Product WHERE maker = 'D' AND type = 'PC')
   UNION ALL
 SELECT price FROM Laptop WHERE model IN (SELECT model FROM Product WHERE maker = 'D' AND type = 'Laptop')
) AS Prices;
```
Evaluation: (1000 + 1100) / 2 = 1050.

e) Find, for each different speed, the average price of a PC.
```sql
SELECT speed, AVG(price) AS avg_price
FROM PC
GROUP BY speed;
```
Evaluation
Speed 2.2: 800
Speed 3.0: 1200
Speed 2.8: 1000

f) Find for each manufacturer, the average screen size of its laptops.
```sql
SELECT P.maker, AVG(L.screen) AS avg_screen_size
FROM Product P
JOIN Laptop L ON P.model = L.model
WHERE P.type = 'Laptop'
GROUP BY P.maker;
```
Evaluation:

Maker A: 15
Maker D: 14.5

g) Find the manufacturers that make at least three different models of PC.
```sql
SELECT maker
FROM Product
WHERE type = 'PC'
GROUP BY maker
HAVING COUNT(DISTINCT model) >= 3;
```
Evaluation: No manufacturers in the sample data meet this condition.

h) Find for each manufacturer who sells PC ’s the maximum price of a PC.

```
SELECT P.maker, MAX(PC.price) AS max_price
FROM Product P
JOIN PC ON P.model = PC.model
WHERE P.type = 'PC'
GROUP BY P.maker;
```

Evaluation:

Maker A: 1200
Maker D: 1000

i) Find, for each speed of PC above 2.0, the average price.

```
SELECT speed, AVG(price) AS avg_price
FROM PC
WHERE speed > 2.0
GROUP BY speed;
```

Evaluation:

Speed 3.0: 1200
Speed 2.8: 1000

j) Find the average hard disk size of a PC for all those manufacturers that make printers

```
SELECT AVG(PC.hd) AS avg_hd_size
FROM PC
WHERE model IN (SELECT model FROM Product WHERE maker IN (SELECT maker FROM Product WHERE type = 'Printer'));
```

Evaluation: (500) / 1 = 500.

Exercise 6.5.2

Write the following database modifications, based on the database schema
Classes(class, type, country, numGuns, bore, displacement)
Ships(name, class, launched)
Battles(name, date)
Outcomes(ship, battle, result)
of Exercise 2.4.3. Describe the effect of the modifications on the data of that exercise.

a) The two British battleships of the Nelson class — Nelson and Rodney — were both launched in 1927, had nine 16-inch guns, and a displacement of 34,000 tons. Insert these facts into the database.

```
INSERT INTO Ships (name, class, launched)
VALUES
   ('Nelson', 'Nelson', 1927),
   ('Rodney', 'Nelson', 1927);
INSERT INTO Classes (class, type, country, numGuns, bore, displacement)
VALUES
   ('Nelson', 'battleship', 'UK', 9, 16 * 2.5, 34,000 * 1.1);
```

Effect: Two British battleships, Nelson and Rodney, of the Nelson class, have been added to the Ships table, and their class information is added to the Classes table.

b) Two of the three battleships of the Italian Vittorio Veneto class — Vittorio
Veneto and Italia — were launched in 1940; the third ship of that
class, Roma, was launched in 1942. Each had nine 15-inch guns and a
displacement of 41,000 tons. Insert these facts into the database.

```
INSERT INTO Ships (name, class, launched)
VALUES
    ('Vittorio Veneto', 'Vittorio Veneto', 1940),
    ('Italia', 'Vittorio Veneto', 1940),
    ('Roma', 'Vittorio Veneto', 1942);
INSERT INTO Classes (class, type, country, numGuns, bore, displacement)
VALUES
    ('Vittorio Veneto', 'battleship', 'Italy', 9, 15 * 2.5, 41,000 * 1.1);
```

Effect: Three Italian battleships, Vittorio Veneto, Italia, and Roma, of the Vittorio Veneto class, have been added to the Ships table, and their class information is added to the Classes table.
c) Delete from Ships all ships sunk in battle.

```
DELETE FROM Ships
WHERE name IN (SELECT ship FROM Outcomes WHERE result = 'sunk');
```

Effect: All ships that have been marked as "sunk" in the Outcomes table have been deleted from the Ships table

d) Modify the C lasses relation so that gun bores are measured in centimeters
(one inch = 2.5 centimeters) and displacements are measured in metric
tons (one metric ton = 1.1 tons).

```
UPDATE Classes
SET bore = bore * 2.5, displacement = displacement * 1.1;
```

Effect: The bore measurements (gun bores) are updated to centimeters (multiplied by 2.5), and displacement measurements are updated to metric tons (multiplied by 1.1) in the Classes table.

e) Delete all classes with fewer than three ships.

```
DELETE FROM Classes
WHERE class IN (SELECT class FROM Ships GROUP BY class HAVING COUNT(*) < 3);
```

Effect: All classes that have fewer than three ships have been deleted from the Classes table.

Exercise 7.5.2

Write the following as triggers. In each case, disallow or
undo the modification if it does not satisfy the stated constraint. The database
schema is from the "PC" example of Exercise 2.4.1:
Product(m aker, model, type)
PC(model, speed, ram, hd, p ric e)
Laptop(m odel, speed, ram, hd, screen , price)
Printer(model, color, type, p rice)

In SQL, triggers are typically used to enforce constraints or perform actions in response to certain events, such as INSERT, UPDATE, or DELETE operations. Below are triggers for each of the specified constraints:

a) When updating the price of a PC, check that there is no lower priced PC
with the same speed.

```
CREATE TRIGGER CheckPCPrice
BEFORE UPDATE ON PC
FOR EACH ROW
BEGIN
  IF NEW.price < (SELECT MIN(price) FROM PC WHERE speed = NEW.speed AND model != NEW.model) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Price cannot be lower than the lowest price for PCs with the same speed.';
  END IF;
END;
```

b) When inserting a new printer, check that the model number exists in
Product.

```
CREATE TRIGGER CheckPrinterModel
BEFORE INSERT ON Printer
FOR EACH ROW
BEGIN
  IF NEW.model NOT IN (SELECT model FROM Product) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Model number does not exist in the Product table.';
  END IF;
END;
```

c) When making any modification to the Laptop relation, check that the
average price of laptops for each manufacturer is at least $1500.

```
CREATE TRIGGER CheckLaptopAveragePrice
AFTER INSERT OR UPDATE ON Laptop
FOR EACH ROW
BEGIN
  IF (SELECT AVG(price) FROM Laptop WHERE maker = NEW.maker) < 1500 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Average price of laptops for this manufacturer is less than $1500.';
  END IF;
END;
```

d) When updating the RAM or hard disk of any PC, check that the updated
PC has at least 100 times as much hard disk as RAM.

```
CREATE TRIGGER CheckPCRAMandHD
BEFORE UPDATE ON PC
FOR EACH ROW
BEGIN
  IF NEW.hd < 100 * NEW.ram THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Hard disk size must be at least 100 times the RAM size.';
  END IF;
```

```
END;
```

e) When inserting a new PC, laptop, or printer, make sure that the model number did not previously appear in any of PC, Laptop, or Printer.

```
CREATE TRIGGER CheckModelUniqueness
BEFORE INSERT ON PC, Laptop, Printer
FOR EACH ROW
BEGIN
   IF NEW.model IN (SELECT model FROM PC UNION ALL SELECT model FROM Laptop UNION ALL SELECT
model FROM Printer) THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Model number already exists in the database.';
   END IF;
END;
```