Database-Systems: Assignment.

Name: Rutvik Barbhai
Reg No: 225805222
Roll No. 39
semester: IV
subject: database-management systems

Q1. (A): There are various categories of applications using database concepts like:-

a) Business and commerce:- (i) 'Customer Relationship Managents (CRM)' store information and track interactions.

ii) 'Inventory Management' tracks inventory levels, orders and deliveries

b) Health care:- (i) 'Electronic Health Records (EHR)' store patient records and medical histories

ii) 'Pharmacy Management' manages drug inventory and preseceptions.

c) Education:- i) 'Student Information system (SIS)' manages student records, grades and attendance

iii) 'Learning Management System (LMS)' stores course materials and student ~~progra~~ progress. and many more.

Q1 B) R x S (Cartesian Product)

A	B	C.X	D.X	C.y	D.y	E
1	2	3	4	1	2	4
1	2	3	4	3	4	1
1	2	3	4	5	1	6
1	2	3	4	4	2	3
2	2	5	1	1	2	4
2	2	5	1	3	4	1
2	2	5	1	5	1	6
2	2	2	1	4	2	3
3	4	2	6	1	2	4
3	4	2	6	3	4	1
3	4	2	6	5	1	6
3	4	2	6	4	2	3
4	2	5	3	1	2	4
4	2	5	3	3	4	1
4	2	5	3	5	1	6
4	2	5	3	4	2	3

Rutvik Barbhai

ii) $R \bowtie_{R.D = S.P}$ (Natural join)

A	B	C.x	P	C.y	E
1	2	3	4	3	1
2	2	5	1	5	6

iii) $R \bowtie_{R.A = S.E} S$ (Natural Join)

A	B	C.x	D.x	C.y	D.y	E
1	2	3	4	3	4	1
3	4	2	6	4	2	3
4	2	5	3	1	2	4

iv a) R ∪ S (union)

A	B	C	D	E
1	2	3	4	–
2	2	5	1	–
3	4	2	6	–
4	2	5	3	–
–	–	1	2	4
–	–	3	4	1
–	–	5	1	6
–	–	4	2	3

b) R ∩ S (Intersection)

A	B	C	D	E
1	2	3	4	1
2	2	5	1	6

v) a) R − S (Difference)

A	B	C	D
3	4	2	6
4	2	5	3

b) S − R (Difference)

C	D	E
1	2	4
4	2	3

Q1c) To analyze whether the given schedules can be converted into a serializable schedule, we need to check if the transactions T_1 and T_2 can be reordered in such a way that final result is equivalent to some serial execution of these transactions.

Given: T_1 : Read (A)

A = A - 50

B = B - 10

write (B)

commit

T_2 : Read (A)

A = A + 10

write (A)

commit

Steps to Analyze serializability :-

i) Identify conflicting operations :- They occur when two transactions access the same data item and at least one of the acesses is a write Here both T_1 and T_2 access A

ii) check for conflicts → T_1 reads A and then writes B
→ T_2 reads A and writes A

iii) conflict Analysis : → T_1 reads A before T_2 reads A
→ T_2 writes A after T_1 reads A

Now to determine serializability, we construct a conflict graph.

→ Nodes represent transactions. Here nodes are T_1 & T_2.

→ Directed edges represents conflicts. Here edge is $T_1 \rightarrow T_2$ because from T_1 to T_2 because T_1 reads A before T_2 writes A

As conflict graph has no cycles, S is conflict serializable.

[Equivalent serial schedule is T_1 followed by T_2].

Q2A)

B+ Tree	B Tree
a) All values are stored at the leaf nodes. Internal nodes only store keys.	a) values can be stored at both internal and leaf nodes
b) Leaf nodes are linked, allowing efficient sequential access and range queries.	b) Leaf nodes are not linked, making sequential access less efficient.
c) It has a lower height for the same number of keys, resulting in faster research times.	c) It may have a higher height, leading to more disc I/O operations during searches
d) It has better space utilization as all keys are stored in the leaf nodes	d) space utilization can be less efficient as values are stored both in internal and leaf nodes.
e) ~~Inser~~ Insertion and deletion is more efficient due to the structure of the tree and linked leaf nodes.	e) Insertion and deletion can be more complex and less efficient as values need to be manged at multiple leads.

Q2B)
```
CREATE TABLE student (
    student_ID INT PRIMARY KEY,
    student_Name VARCHAR2(255),
    Age INT,
    Hobby VARCHAR2(255)
    DOB DATE,
```

```
Address-ID INT,
FOREIGN KEY (Address-ID) REFERENCES Address
(Address-ID)
);

CREATE TABLE course(
    course-ID INT PRIMARY KEY,
    course-Name VARCHAR2(255)
    );

CREATE TABLE subjects(
    Subject-ID INT PRIMARY KEY,
    Subject-Name VARCHAR2(255),
    LECTURER-ID INT,
    FOREIGN KEY (LECTURE-ID) REFERENCES
    LECTURER (Lecturer-ID)
```

Q2C) i)

```
SELECT DISTINCT suppliers.sid
FROM suppliers
JOIN catalog ON suppliers.sid = catalog.sid
JOIN parts ON catalog.pid = parts.pid
WHERE parts.colour = 'red' AND
suppliers.address = '221 packer street';
```

ii)

```
SELECT DISTINCT suppliers.sid
FROM suppliers
JOIN catalog ON suppliers.sid = catalog.sid
JOIN parts ON catalog.pid = parts.pid
WHERE parts.colour = 'red';
```

Q3A) i)

```
SELECT Appointment.time, Clients.name
FROM Appointments
JOIN staff ON Appointment.SID = staff.SID
JOIN client ON Appointment.CID = clients.CID
```

```
WHERE staff.name = 'Amar' AND MONTH(
  Appointment.date) = 6;
```

ii)
```
SELECT DISTINCT clients.CID, clients.name,
  clients.phone
  FROM Clients
  JOIN Appointment AS A1 ON Clients.CID = A1.CID
  JOIN Appointment AS A2 ON Client.CID = A2.CID
WHERE A1.service = 'order processing' AND
  A2.service = 'Technical support';
```

iii)
```
SELECT DISTINCT staff.name FROM staff

  JOIN Appointments on staff.SID = Appointment.SID
  JOIN clients ON Appointments.CID = clients.CID
    WHE = clients.name LIKE 'A%-h'
    ORDER BY staff.name DESC;
```

iv)
```
SELECT DISTINCT clients.name AS client_Name
  FROM clients
  JOIN Appointment AS A1 ON Clients.CID = A1.CID
  WHERE A1.date > (SELECT MIN(A2.date)
                   FROM Appointments AS A2
                   WHERE A2.service = 'Technical
                                      support'
                   );
```

v)
```
SELECT DISTINCT Clients.CID, clients.name
  FROM clients
JOIN Appointments ON Clients.CID = Appointments.CID
JOIN staff ON Appointments.SID = staff.SID
WHERE Appointments.service = 'order processing'
  AND client.name <> staff.name;
```

Q3 B) A functional dependency $X \rightarrow Y$ is non-trivial if Y is not a subset of X.

→ Analysis => $A \xrightarrow{*} B$: For each unique value of A, B is always b1 : This is non trivial F.D

=> $B \rightarrow A$ For each unique value of B, A can be a_1 or a_2, This is not a valid F.D

=> $A \rightarrow C$: For each unique value of A, C can be c_1 or c_2 for a_1 and c_1 or c_3 for a_2. This is not a valid F.D

=> $B \rightarrow C$: For each unique value of B, C can be c_1, c_2 or c_3. This is not a valid F.D.

=> $A, B \xrightarrow{*} C$: For each unique combination of A and B, C is uniquely determined. This is a valid non-trivial F.D

=> $C \rightarrow A$: For each unique value of C, A can be a_1 or a_2. This is not a valid F.D

=> $C \xrightarrow{*} B$: For each unique value of C, B is always b_1. This is a valid non-trivial F.D

Q3 C)

i) SELECT S.name AS staff_Name, A.service, COUNT(DISTINCT A.CID) AS NO OF Distinct clients
FROM staff S
JOIN Appointments A on S.SID = A.SID
GROUP By S.name, A.service
ORDER By S.name, A.service

ii) DELECT FROM staff
WHERE SID NOT IN (SELECT DISTINCT SID FROM Appointments).

Q4 A) i) a Natural join:-

ID	Name	Block-No	Room-No
1	Abhi	AB5	215
2	Adam	AB1	802
3	Alex	AB5	316

b) Left outer join

ID	Name	Block-No	Room-No
1	Abhi	AB5	215
2	Adam	AB1	302
3	Alex	AB5	316
4	Anu	—	—

c) Full outer join

ID	Name	Block-No	Room-No
1	Abhi	AB5	215
2	Adam	AB1	302
3	Alex	AB5	816
4	Anu	—	—

ii) Referential integrity ensures that the relationships between tables remain constaint. It means that a foreign key in one table must always refer to a valid primary key in another table

To handle the violation given, we can use ON DELETE CASCADELESS with foreign key constraint.

Q48) To design the database in 3NF, we need to ensure that:-

(i) The relation is in 2NF

ii) There are no transiture dependencies

so we assume:-

- Each department has a unique department number
- Each employee has a unique employee number
- Each project has a unique project numbers.
- Each department has a unique manager.
- Each employee is assigned to one project at a time.

Relations:-

1) Department :- a) Attributes :- Dept No (PK), Budget ManagerEmployeeNumber (FK)

b) Dependencies → Dept No → Budget, Manager Employee Number

2) Employee - a) Attributes - EmpNo (PK), CurProj No (FK) OfficeNo, Phone No.

b) Dependencies :- EmpNo → CurProj No, OfficeNo, Phone No.

3) Projects :- a) Attributes :- ProjNo (PK), Budget

b) Dependencies :- ProjNo → Budget

4) DeptProjects :- a) Attributes :- Dept No (FK), Proj No (F.K)

b) Dependencies :- Dept No, ProjNO → (No of additional attributes)

⇒ This design ensures that the database is 3NF by eliminating transitive dependencies & ensuring that each non key attribute is fully functionally dependent on the primary key.

This design also maintains referential integrity through the use of foreign keys.

Q4C) Relational schema:

a) Scientist
```
CREATE TABLE scientist (
    sid INT PRIMARY KEY,
    Sname VARCHAR2(255),
    Fname VARCHAR2(255),
    Lname VARCHAR2(255),
    country VARCHAR2(255),
);
```

b) Research Area :
```
CREATE TABLE ResearchArea(
    SID INT,
    RArea VARCHAR2(255)
    PRIMARI KEY (sid, RArea),
    FOREIGN KEY (sid) REFERENCES
    scientist(sid)
);
```

Invention :
```
CREATE TABLE Invention(
    IID INT PRIMARY KEY,
    Iname VARCHAR2(255),
    Year INT
);
```

d) Invents :
```
CREATE TABLE Invents (
    sid INT,
    IID INT,
    PRIMARY KEY (sid, IID);
    FOREIGN KEY (sid) REFERENCES
    Scientist(sid),
    FOREIGN KEY (IID) REFERENCES
    Inventor(IID)
);
```

Q 5 A) Given Functional Dependencies : $A \rightarrow BCD$
$BC \rightarrow DE$
$B \rightarrow D$
$D \rightarrow A$

i) To compute B+, we apply FDs iteratively until no new attributes can be added.

→ start with $B+ = \{B\}$
→ using $B \rightarrow D$, add D to $B+ = \{B, D\}$
→ using $D \rightarrow A$, add A to $B+ = \{B, D, A\}$
→ using $A \rightarrow BCD$, add C to $B+$: $B+ = \{B, D, A, C\}$
→ using $BC \rightarrow DE$, add E to $B+$: $B+ = \{B, D, A, C, E\}$

ii) To prove AF is a superkey, we need to show that $(AF)+$ contains all attributes of R.

→ start with $(AF)+ = \{A, F\}$
→ using $A \rightarrow BCD$, add B, C, D to $(AF)+$: $(AF)+ = \{A, F, B, C, D\}$
→ using $BC \rightarrow DE$, add E to $(AF)+$: $(AF)+ = \{A, F, B, C, D, E\}$

$(AF)+$ contains all attributes of R:
$(AF)+$ is a superkey

iii) Compute canonical cover

a) Decompose FDs so that each has a single attribute on RHS:-

$A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $BC \rightarrow D$, $BC \rightarrow E$. $B \rightarrow D$. $D \rightarrow A$.

b) Remove extraneous attributes from LHS:-

$A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $BC \rightarrow E$, $B \rightarrow D$. $D \rightarrow A$

c) Remove redundant FDs:-

$A \rightarrow B$. $A \rightarrow C$, $A \rightarrow D$ $BC \rightarrow E$. $B \rightarrow D$. $D \rightarrow A$

final canonical cover

iv) 3NF decomposition

using canonical cover, we decompose R into 3NF relations:-

R_1 (A, B, C, D) with $A \to B$, $A \to C$, $A \to D$

R_2 (B C, E) with $BC \to E$

R_3 (B, D) with $B \to D$

R_4 (D, A) with $D \to A$.

v) BCNF decomposition:-

a) Identify violations of BCNF:-

$A \to BCD$ (violates BCNF if A is not a superkey)

$BC \to DE$ (violates BCNF if BC is not a superkey)

$B \to D$ (violates BCNF if B is not a superkey)

$D \to A$ (violate BCNF if D is not a superkey)

b) Decompose based on violations:-

Decompose R into R_1 (A, B, C, D) and R_2 (D, A)

Decompose R_1 into R_3 (A, B, C) and R_4 (B, D)

Decompose R_3 into R_5 (A, B) and R_6 (BC, E)

→ Final BCNF relations:-

1. R_1 (A, B)
2. R_2 (D, A)
3. ~~R_3 (C, D)~~ R_3 (B, D)
4. R_4 (BC, E)
5. R_5 (A, C)

Q5B)

$A \rightarrow D$ (derived from $A \rightarrow B$ and $B \rightarrow D$)

$E \rightarrow BC$ (derived from $E \rightarrow A$ and $A \rightarrow BC$)

$E \rightarrow D$ (derived from $E \rightarrow A$ and $A \rightarrow D$)

$BC \rightarrow E$ (derived from $B \rightarrow D$, augmented to $BC \rightarrow CD$ and $CD \rightarrow E$)

$BE \rightarrow A$ (derived through augmentation & transitivity

$BE \rightarrow DC$ (derived through augmentations and transitivity)

Q5C)

Given T13 and T14 and $A = 0$ and $B = 0$

→ Consistency requirement :- $A = 0$ V $B = 0$

Non-serializable schedule:

→ T13 reads A : $A = 0$

→ T14 reads B : $B = 0$

→ T13 reads B : $B = 0$

→ T14 reads A : $A = 0$

→ T13 checks if $A = 0$: True so $B = B + 1 \rightarrow B = 1$

→ T14 checks if $B = 0$: True so $A = A + 1 \Rightarrow A = 1$

→ T13 writes B : $B = 1$

→ T14 write A : $A = 1$

Final values: $A = 1$, $B = 1$ and it violates the consistency requirement. Therefor it proves this schedule is non-serializable