Task 7: Triggers, views and Exceptions

Aim:- To conduct events, views, exceptions on CRUD operations for restricting phenomenon

a) To creat a trigger in PL/SQL that automatically inserts a new record in the match_result table when a new record is inserted into the match table

b) To create a view that displays the details of players along with their team details.

c) To write a non-recursive PL/SQL procedure to retrieve even-numbered player ID registered for any match with exception handling

a).

```
SQL> CREATE TABLE MATCH_Result (ResultID
varchar(10) primary KEY, MATCHID VARCHAR
(10), TEAMID varchar(0), Result varchar(20)
FOREIGN KEY (matchID) REFERENCES
MATCH (MATCHID), FOREIGN KEY (TeamID) REFERENCES team(TeamID));

SQL> CREATE OR REPLACE TRIGGER trg_
insert_match_result AFTER INSERT ON MATCH
    FOR EACH ROW
    DECLARE
      N NUMBER;
    BEGIN
      N:= INSTR (UPPER(:New.RESULT), 'WIN');
    IF N!= AND INSTR(UPPER(:NEW.TEAMID))
    THEN
    INSERT INTO MATCH_Result (ResultID, MATCH
      , TEAMID, Result)
VALUES ('R'||:NEW.MATCHID||'1';:NEW.MATCH_ID,
                    TEAMID, Result)
INSERT INTO MATCH_Result (ResultID, MATCH
    ID, TEAMID, Result)
VALUES ('R'||:NEW.MATCH||'2';NEW.TEAM
                    ID2, 'FAIL');
```

```
ELSE
INSERT INTO MATCH_Result (ResultID, MATCHID,
TEAMID, Result)
VALUES ('R'||:NEW.MATCHID||'1', :NEW.MATCHID,
:NEW.TEAMID1, 'WIN');
INSERT INTO Match_Result (ResultID, MATCHID,
TEAMID, Result)
VALUES ('R'||:NEW.MATCHID||'2', :NEW.MATCHID,
:NEW.TEAMID2, 'WIN');
END IF;
END;
```

```
SQL> Insert into match values ('M01', 'CCBO1', 'TCBO1', '22-JUN
-2022', 1.3, 'TCBO1-WIN');

SQL> Insert into match values ('M02', 'CCBO2', '22-JUN
-2022', 8.3, 'CCBO1-WIN');

SQL> Insert into match values ('M03', 'TRICBO1', '22-JUN
-202', 8.3, 'TCBO1-WIN');

SQL> Insert into match values ('M04', 'TRICBO1', 'TC
BO2', '25-JUN-2022', 8.3, 'TRICBO1-WIN');

SQL> Insert into match values ('M05', 'TCBO1', 'TCBO2',
DATE '2025-04-20', 16.00, 'TCBO2 WIN');

SQL> select * from match_Result;
```

b. To create a view that displays the details of players along with their team details.

```
SQL> CREATE OR REPLACE VIEW player_Team view
AS SELECT P.playerID, P.FNAME AS First
NAME, P.Playing Role, P.Batting AS Batting
style, P.Bowling AS Bowling style, t.team
ID, t.tname AS team Name, t.coach AS
coach, t.captain AS captain FROM player
P JOIN team t ON P.TEAM ID = t.teamID
```

5a) select * from player_team_vicw;

6. To write a non-recursive PL/SQL Procedure to retrieve even-numbered player IDs registered for any match with exception handling

```
CREATE OR REPLACE PROCEDURE Get_Even_
  player_IDs_For_matches_Is
    cursor even_players_ID
    SELECT DISTINCT p.player_ID, P.FName, P.LN
    CM.matchID
    from player p
    JOIN team ON p.team_ID = t.teamID
    Join match m ON (m.team_ID1 = t.teamID
    OR m.teamID2 = t.teamID)
    WHERE MOD (TO_NUMBER (REGEXP_
    SUBSTR(p.player_ID, '\d+'), 2) = 0;
  v_count NUMBER := 0;
    BEGIN
  DBMS_output.put_LINE ('Even-number
  player IDs registered for matches:');
  FOR rec IN even_players_cur
    v_count := v_count + 1;
  DBMS_output_LINE ('matchID:' || rec.
    MATCH_ID ||
    ', playerID:' || rec.player_ID ||
    ', Name:' || rec.FName || ' ' || rec.LName);
  END cur;
  -- Handle case where no records found
  IF v_count = 0 THEN
    RAISE NO_DATA_FOUND;
    END IF;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
  DBMS_output_LINE ('No even number
  player IDs found for any match.');
```

WHEN OTHERS THEN
DBMS_output.put_line('Error: '||SQLERRM);
END;
/

SQL> set serverout on
SQL> /
Procedure created.

SQL> SET SERVEROUT ON;
SQL> EXEC Get_Even_Player_EOS_For_Matches;

VEL TECH - CSE

EX NO.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (3)	
VIVA VOCE (3)	
RECORD (4)	
TOTAL (15)	
SIGN WITH DATE	

Result:- Thus the triggers, views and exceptions experiment was successfully completed results are verified

Assignments - I, II

course code : 10211CS207

course Name : Data base management system

SLOT :- S1L4

Faculty : Dr. N. Rajkumar sir.

course category :- programcore

Name : K. Rakesh

VTU : 30036

Branch : CSE(AI/ML)

30/10/25

NO	Questions	CO - NO	K-Lev
	TASK -4		
1	HOSPITAL MANAGEMENT SYSTEM Many doctors, nurses, and staff are working together in a hospital to manage various patient records and treatment processes. The hospital ownership may either be self-owned or operated under a trust or corporate consortium. Each patient visits the hospital for consultation, diagnosis, and treatment, and their details must be properly maintained and updated. As per the requirement, collect and maintain the following details: • Patient ID • Patient name, age, and gender • Doctor name and specialization • Appointment date and time • Department and room number • Diagnosis and prescribed medicines • Treatment cost and availability of medical facilities • Patient contact number and address • Admission and discharge date, treatment status • Bill details including payment status (paid/unpaid), date, and insurance information Tasks: I. Construct an Entity–Relationship (ER) Diagram for the above Hospital Management System.	CO1	K3
2	II. Design a database based on the ER diagram and implement it using SQL to perform appropriate operations for the given task.	CO2	K3
3	Consider the relation $R = \{A, B, C, D, E\}$ and the following set of functional dependencies: F={AB→C,A→D,C→E,D→F}. Tasks: 5. Find the candidate key(s) for R. 6. Decompose R into Second Normal Form (2NF) relations. Decompose R into Third Normal Form (3NF) relations.	CO3	K3

S.NO	Questions	CO - NO	K-Level
	TASK -4		
1	Consider two transactions, T0 and T1, where Transactions T0 and T1 operate on P, Q and R. Initial values: P=300, Q=400, R=500 respectively. The operations of the transactions are as follows: T0: Read(P); P := P + 50; Write(P); Read(Q); Q := Q - 50; Write(Q); T1: Read(R); R := R - 25; Write(R); Specify these transactions in the log and show the state of the database for both deferred database modification and immediate database modification approaches.	CO4	K2
2	Identify the various use cases in the Company Database where challenges have been turned into opportunities to better serve employees using a NoSQL database.	CO5	K3

Entities and attributes

1. Patient
   - Patient ID (primary key)
   - Name
   - Age
   - Gender
   - Contact Number
   - Adress
   - Insurance Information
2. Doctors
   - Doctor ID (primary key)
   - Name
   - specialization
   - Department
   - RoomNumber
3. Appointment
   - Appointment ID (primary key)
   - Appointment Date
   - Appointment Time
   - Treatment status
4. Diagnosis
   - Diagonasis ID (primary key)
   - Diagnosis Details
   - prescribed Medicines

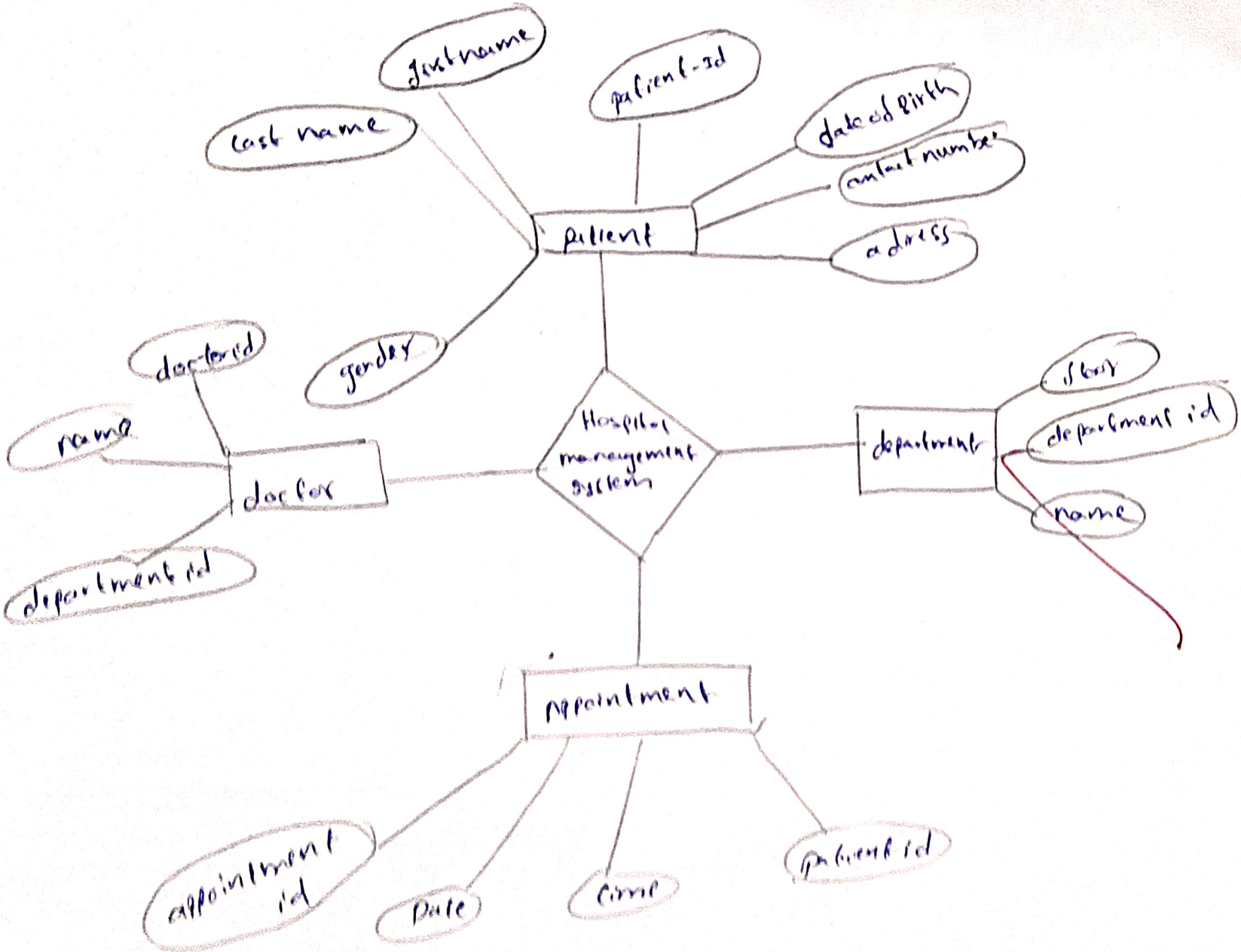
Bill

- Bill ID (primary key)
- Payment - status (paid/unpaid)
- Bill Date
- treatment cost

6. Department
   - Department - ID (primary key)
   - Department name
   - Availibility of medical facilities.

Relationship

1. Patient - appointment
   - one patient can have many appointments
   - Relationship : one - to - many (1:N)
   - Foreign key: patient ID in Appointment
2. Doctor - Appointment
   - one doctor can have many appointments
   - Relationship : one - to - many (1:N)
   - foriegn key : Doctor ID in Appointment
3. Appointment - Diagnosis
   - one appointement can have one or more diagonses.
   - Relationship: one - to - one (1:1) or one - to - many (1:N)
4. patient - Bill
   - one patient can have multiple bills
   - Relationship : one - to - many (1:N)
   - foriegn key : patient ID in Bill


first name
last name
patient-id
date of birth
contact number
address
gender
patient
doctor id
name
doctor
department id
Hospital management system
department
if boy
department id
name
appointment
appointment id
Date
time
patient id

create Database

CREATE DATABASE Hospital management

USE Hospital management;

1. patient table

CREATE TABLE patient (

patient ID INT PRIMARY KEY AUTO-INCREMENT

Name VARCHAR(6),

Age INT.

Gender VARCHAR(6),

Contact Number VARCAR (15),

Adress VARCHAR (200),

Insurance Information VARCAR(6c)

2. Department table

CREATE TABLE Department (

Department ID INT primary key AUTO-Increment,

Department Name VARCHAR (100),

Availibility of medical facilities VARCHAR (200)

);

3. Doctor table

CREATE table Doctor (

Doctor ID INT. PRIMARY KEY AUTO-INCREMENT

Name VARCHAR(100),

specialization VARCHAR(100),

Room Number VARCHAR(6),

Department ID INT,

FOREIGN key (Department ID) REFERENCES

Department (Department ID)

);

Bill Table:-

```
CREATE TABLE BILL(
BILLID INT PRIMARY KEY AUTO- INCREMENT
  BILL DATE,
  Payment status VARCHAR(20),
  treatment cost DECIMAL(10,2)
    patient ID INT
  FOREIGN KEY (patient ID) REFERENCES patient
  (Patient ID)
    );
```

sample sql operations

Insert Data

```
INSERT INTO patient (Name, Age, Gender, contact
    Number, Adress, )
VALUES ('John smith', US, 'mald 9876543210', New
    York', 'Aetna Health');

INSERT INTO Doctor (Name, specialization, Room
Number, Department ID)
  VALUES ('Dr. Alice Brown', 'cardiologist', '203', 1);

INSERT Appointment (Appointment Date, Appointment time,
  treatment status, Patient ID Doctor ID)
    VALUES ('2025-10-28', 10:00:00', ongoing', 1);

INSERT INTO Diagonesis (Diagonosis, Details, prescribed
  medicines, Appointment ID)
      VALUES ('Hypertension', 'Lensartan', 1);

INSERT INTO Bill (Bill-Date, Payment status, treat
  ment cost, Patient ID)
    VALUES ('2025-10-29', 'paid', 5000.00, 1);
```

are transitive chains AB→C → also AB→
nsitively; also C→E → Digives C→D to get a 3NF
relations for the minimal cover (are relation
er FD left in

1. From AB→C → R_ABC (A, B, C) (key: AB)
2. From A→D → R_AD (A, D) (key: A)
3. From C→E → R_CE (C, E) (key: C)
4. From E→D → R_ED (E, D) (key: E)

a dependency - preserving | Loss less 3NF

composition is:

R_1 (A, B, C) - enforce AB→C
R_2 (A, D) - enforce A→D
R_3 (C, E) - enforce C→E
R_4 (E, D) - enforce E→D

candidate key(cc) for R(A, B, C, D, E)

compute closures under F:

- $A^+ = \{A, D\}$ (from $A \to D$)
- $B^+ = \{B\}$
- $(AB)^+ : AB \xrightarrow{AB \to C} C \xrightarrow{C \to E} E$ $E \to D$. So $(AB)^+ =$ $\{A, B, C, E, D\}$ i.e. all attributes

candidate keys (CK): {A, B}

Decompose R into second Normal form (2NF)

Definition reminder: 2NF removes partial dependencies of a non-prime attribute on part of a candidate key: the candidate key is AB. we have partial dependency:

- $A \to D$ → attribute A (part of key AB) determining non-prime attribute D ⇒ partial dependency.

to remove it, separate the relation into:

- $R_1(A, D)$ → implement $A \to D$ (primary key: A)
- $R_2(A, B, C, E)$ - contains the rest (keeps AB as key and the FD $AB \to C$, plus $C \to E$).

$R_1(A, D)$

$R_2(A, B, C, E)$

Decompose R into third Normal form (3NF)

Goal:- remove transitive dependencies (and satisfy 3NF condition). Look at nontrivial

FDs:

- $AB \to C$
- $A \to D$

there are transitive chains AB→C → also AB → transitively; also C→E → D gives C→D to get a 3NF, form relations for the minimal cover (one relation per FD left in

1. From AB→C → R_ABC (A, B, C) (key: AB)
2. From A→D → R_AD (A, D) (key: A)
3. From C→E → R_CE (C, E) (key: C)
4. From E→D → R_ED (E, D) (key: E)

so a dependency - preserving / lossless 3NF decomposition is:

R_1 (A, B, C) - enforce AB→C

R_2 (A, D) - enforce A→D

R_3 (C, E) - enforce C→E

R_4 (E, D) - enforce E→D

Given:

- Initial values:

P = 300, Q = 400, R = 500

- Transactions

T0 = 300, Q = 400, R = 500

T0: Read (P)
P := P + 50
write (P)
Read (Q)
Q := Q - 50
write (Q)

T1: Read (R)
R := R - 25
write (R)

a) Defered database modification

step	operation	Action	Database values (P, Q, R)
			300, 400, 500
1	P := P + 50	T0 reads P = 300	300, 400, 500
2	Read (Q)	T0 updates P = 350	300, 400, 500
3	Q := Q - 50	T0 reads Q = 400	300, 400, 500
4	Read (R)	T0 updates Q = 350	300, 400, 500
5	R := R - 25	T1 reads R = 500	3000, 400, 500
6	R := R - 25	T1 R = 475	300, 400, 500
7	T0 commit	Apply P = 350, Q = 350	300, 400, 500
8	T1 commit	Apply R = 475	300, 400, 500

P = 350
Q = 350
R = 475

a) Immediate Database modification

step	operation	T1 (Tx)	Database (P, Q, R)
1	Read (P)	To read P = 250	P=250, Q=350, R=495
2	P := P+50	To P = 250+50	P=300, Q=350, R=495
3	Read (Q)	To read Q = 350	P=300, Q=350, R=495
4	Q := Q-50	To Q = 350-50	P=300, Q=300, R=495
5	Read (R)	T1 reads R=495	P=300, Q=300, R=495
6	R := R-25	T1 → R = 495-25	P=300, Q=300, R=470
7	To commit	confirmed	P=300, Q=300, R=470
8	T1 commit	confirmed	P=300, Q=300, R=470

R = 250
Q = 350
R = 495

) use cases for No SQL in company Database

use case

1. employee attendance Tracking

challenges

Relation DBs struggle with large, real-time time-stamp data

2- employee feedback & surveys

Unstructured feedback data (text, ratings)

3. project collaborations

Frequent updates and tracking of project docs

4. HR analysis

need for fast querying of dynamic data (skills, departments, performance)

5. employee

High-volume chat messages storage