Database Design Project

CS 6360.002

Project Title: Indoor Playground

Names of the students in team:

1. Sai Sandeep Vaddi – SXV160830
2. Sandeep Avula – SXA165930

Step 1: Data Requirements

Indoor Playground has to has to maintain the data regarding the following to maintain successful business

* 1. Playground Branches
  2. Games
  3. Game equipment
  4. Employees
  5. Customers
  6. Billing
  7. Other shopping like toy shops

## **Branches**

Location details of the indoor playground like the following

1. Details of manager of branch
2. Branch working times
3. Branch area, ground type details
4. Address

## **Games**

1. Playground should have following details about games it hosts:
   1. Game names
   2. Game types like single player, multi-player etc
   3. Age limits for each game
   4. Each game price

## **Equipment**

1. Equipment details of each game
2. Available units of each game according to branches
3. Timings of occupancy on the equipment

## **Employees**

1. Employee name, ssn, sex, salary details
2. Employee’s manager details
3. Employee address details

## **Customers**

1. Contact details
2. Games customers played
3. Their payment details
4. Their membership details

## **Other shops like toy shops**

1. Maintain toys details like which customer is buying which toy etc.

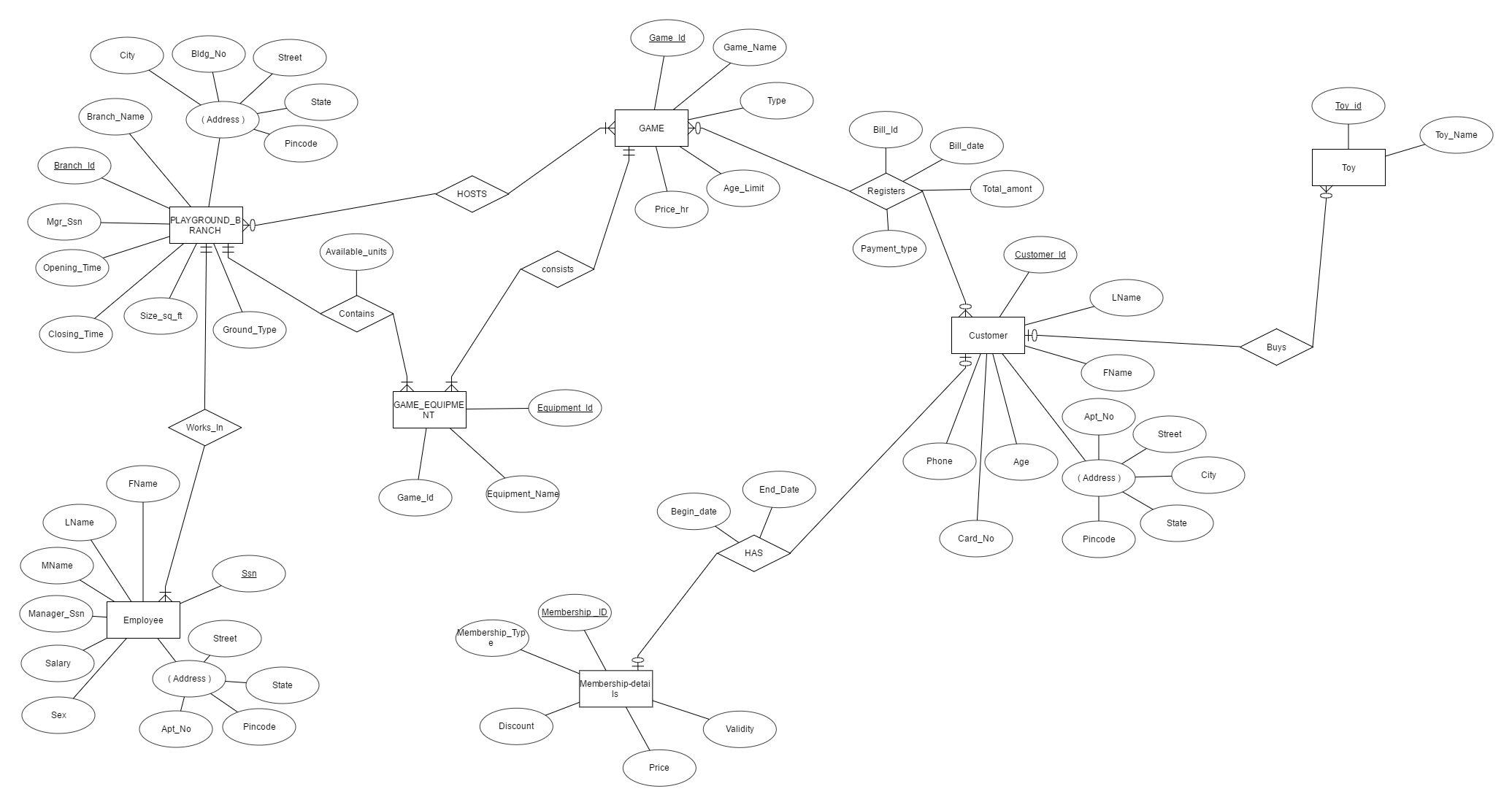
## **Billing details**

1. Which games each customer played
2. How much time they spent on each game
3. Bill date
4. Total amount
5. Final bill with the discount if they have membership

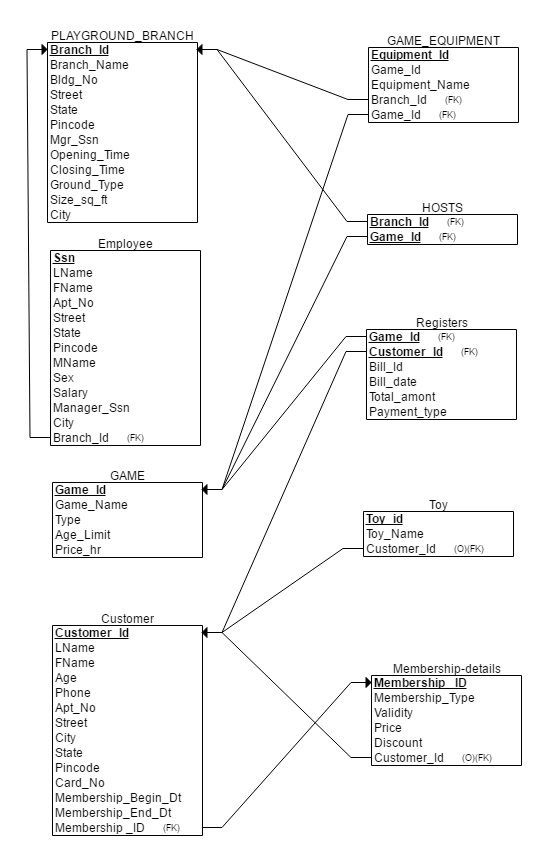
## **Membership**

1. Maintain membership types (Platinum, Gold, Silver)
2. Membership validity
3. Discount available with each membership

Step 2: ER Diagrams



Step 3: ER to Relational Schema



Step 4: Normalization

1)

In PLAYGROUND\_BRANCH, EMPLOYEE, CUSTOMER

City -> State violates 3NF

2)

In CUSTOMER Table, the following FDs violate 3NF

Customer\_Id -> Membership\_Id , Membership\_Begin\_Dt , Membership\_End\_Dt

Membership\_Id , Membership\_Begin\_Dt -> Membership\_End\_Dt

3)

In GAME\_EQUIPMENT table, the following FDs violate 3NF

Equipment\_id -> Equipment\_Name, Game\_Id

Equipment\_Name -> Game\_Id

4)

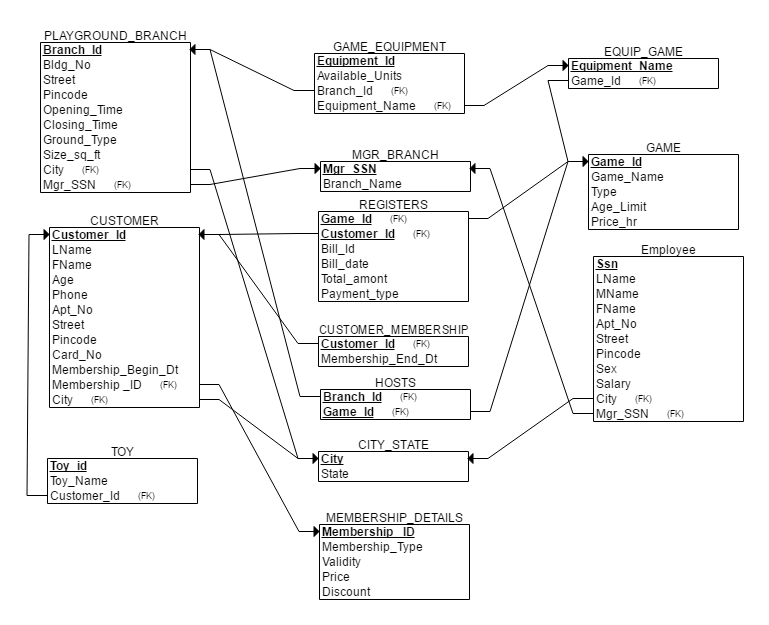
In PLAYGROUND\_BRANCH table, the following FDs violate 3NF

Branch\_Id -> Mgr\_SSN, Branch\_Name

Mgr\_SSN -> Branch\_Name

Branch\_Name -> Mgr\_Ssn

Step 5: Final relational schema after normalization



Step 6: Table creation queries

-- 1.

CREATE TABLE GAME (

Game\_Id INT NOT NULL,

Game\_Name VARCHAR(20) NOT NULL,

Type VARCHAR(20),

Age\_Limit INT,

Price\_Hr DECIMAL(5,2),

PRIMARY KEY (Game\_Id)

);

-- 2.

CREATE TABLE MEMBERSHIP\_DETAILS(

Membership\_Id INT NOT NULL,

Membership\_Type VARCHAR(20) NOT NULL,

Validity INT,

Price DECIMAL(5,2),

Discount INT,

PRIMARY KEY (Membership\_Id)

);

-- 3.

CREATE TABLE CITY\_STATE (

City VARCHAR (20),

State VARCHAR (20),

PRIMARY KEY (City)

);

-- 4.

CREATE TABLE MGR\_BRANCH(

Mgr\_Ssn VARCHAR(9) NOT NULL,

Branch\_Name VARCHAR(20) NOT NULL,

PRIMARY KEY (Mgr\_Ssn)

);

-- 5.

CREATE TABLE PLAYGROUND\_BRANCH

(

Branch\_Id INT NOT NULL,

Mgr\_Ssn VARCHAR(9) NOT NULL,

Opening\_Time TIMESTAMP,

Closing\_Time TIMESTAMP,

Size\_Sq\_Ft VARCHAR(20),

Ground\_Type VARCHAR(20),

Bldg\_No VARCHAR(20),

Street VARCHAR(20),

City VARCHAR(20),

Pincode INT,

PRIMARY KEY (Branch\_Id),

FOREIGN KEY (City) REFERENCES CITY\_STATE (City) ON DELETE SET NULL,

FOREIGN KEY (Mgr\_Ssn) REFERENCES MGR\_BRANCH (Mgr\_Ssn) ON DELETE SET NULL

);

-- 6.

CREATE TABLE HOSTS(

Branch\_Id INT NOT NULL,

Game\_Id INT NOT NULL,

PRIMARY KEY (Branch\_Id, Game\_Id),

FOREIGN KEY (Branch\_Id) REFERENCES PLAYGROUND\_BRANCH (Branch\_Id) ON DELETE CASCADE,

FOREIGN KEY (Game\_Id) REFERENCES GAME (Game\_Id) ON DELETE CASCADE

);

-- 7.

CREATE TABLE CUSTOMER

(

Customer\_Id INT NOT NULL,

Lname VARCHAR(20) NOT NULL,

Fname VARCHAR(20) NOT NULL,

Age INT,

Phone VARCHAR(15),

Membership\_Id INT,

Membership\_Begin\_Dt DATE,

-- Membership\_Status VARCHAR(20),

-- Membership\_End\_Dt Date,

Card\_No VARCHAR(20),

Apt\_No VARCHAR(20),

Street VARCHAR(20),

City VARCHAR(20),

Pincode INT,

PRIMARY KEY (Customer\_Id),

FOREIGN KEY (Membership\_Id) REFERENCES MEMBERSHIP\_DETAILS(Membership\_Id) ON DELETE SET NULL,

FOREIGN KEY (City) REFERENCES CITY\_STATE (City) ON DELETE SET NULL

);

-- 8.

CREATE TABLE TOY(

Toy\_Id INT NOT NULL,

Toy\_Name VARCHAR(20),

Customer\_Id INT,

PRIMARY KEY (Toy\_Id),

FOREIGN KEY (Customer\_Id) REFERENCES CUSTOMER(Customer\_Id) ON DELETE SET NULL

);

-- 9.

CREATE TABLE EQUIP\_GAME (

Equipment\_Name VARCHAR(20) NOT NULL,

Game\_Id INT NOT NULL,

PRIMARY KEY (Equipment\_Name),

FOREIGN KEY (Game\_Id) REFERENCES GAME (Game\_Id) ON DELETE CASCADE

);

-- 10.

CREATE TABLE GAME\_EQUIPMENT (

Equipment\_Id INT NOT NULL,

Equipment\_Name VARCHAR(20),

Branch\_Id INT,

Available\_Units INT,

PRIMARY KEY (Equipment\_Id),

FOREIGN KEY (Equipment\_Name) REFERENCES EQUIP\_GAME (Equipment\_Name) ON DELETE CASCADE,

FOREIGN KEY (Branch\_Id) REFERENCES PLAYGROUND\_BRANCH (Branch\_Id) ON DELETE SET NULL

);

-- 11.

CREATE TABLE REGISTERS(

Customer\_Id INT,

Game\_Id INT,

Bill\_Id INT,

Bill\_Date DATE,

Payment\_Type VARCHAR(20),

Total\_Amount DECIMAL(10,2),

PRIMARY KEY (Customer\_Id, Game\_Id),

FOREIGN KEY (Customer\_Id) REFERENCES CUSTOMER (Customer\_Id) ON DELETE SET NULL,

FOREIGN KEY (Game\_Id) REFERENCES GAME (Game\_Id) ON DELETE SET NULL

);

-- 12.

CREATE TABLE EMPLOYEE (

SSN VARCHAR(9) NOT NULL,

Lname VARCHAR(20) NOT NULL,

Mname VARCHAR(20) NOT NULL,

FName VARCHAR(20) NOT NULL,

Mgr\_Ssn VARCHAR(9),

Salary DECIMAL(10,2),

Sex VARCHAR(20),

Apt\_No VARCHAR(20),

Street VARCHAR(20),

City VARCHAR(20),

Pincode VARCHAR(5),

PRIMARY KEY (SSN),

FOREIGN KEY (City) REFERENCES CITY\_STATE (City) ON DELETE SET NULL,

FOREIGN KEY (Mgr\_Ssn) REFERENCES MGR\_BRANCH (Mgr\_Ssn) ON DELETE SET NULL

);

-- 13

CREATE TABLE CUSTOMER\_MEMBERSHIP (

Customer\_Id INT NOT NULL,

Membership\_End\_Dt DATE,

PRIMARY KEY (Customer\_Id),

FOREIGN KEY (Customer\_Id) REFERENCES CUSTOMER (Customer\_Id)

);

Step 7: Stored Procedures

**1. Stored Procedure to retrieve the customer details whose membership status end by today.**

DROP PROCEDURE Customers\_Mem\_Curr\_Status;

CREATE or replace PROCEDURE Customers\_Mem\_Curr\_Status As

this\_c\_Id CUSTOMER.CUSTOMER\_ID%TYPE;

this\_c\_FName CUSTOMER.FNAME%TYPE;

this\_c\_LName CUSTOMER.LNAME%TYPE;

CURSOR RES\_SET IS

SELECT C.Customer\_Id,C.Fname,C.Lname FROM CUSTOMER C WHERE C.MEMBERSHIP\_END\_DT <= trunc(SYSDATE) and C.MEMBERSHIP\_ID IS NOT NULL

FOR UPDATE;

BEGIN

OPEN RES\_SET;

LOOP

FETCH RES\_SET INTO this\_c\_Id, this\_c\_FName ,this\_c\_LName;

EXIT WHEN(RES\_SET%NOTFOUND);

dbms\_output.put\_line('Customer\_ID = ' || this\_c\_Id ||

', Customer\_Name = ' || this\_c\_FName || ', First\_Name = ' || this\_c\_LName );

END LOOP;

CLOSE RES\_SET;

END;

.

RUN;

BEGIN

Customers\_Mem\_Curr\_Status();

END;

.

RUN;

**2. Stored Procedure for generating Bill for each customer today calculating the changes in the bill amount according to membership discount if they have membership.**

DROP PROCEDURE GENERATE\_BILL; COMMIT;

CREATE OR REPLACE PROCEDURE GENERATE\_BILL (Customer\_Id IN CUSTOMER.Customer\_Id%TYPE)

IS

Gid GAME.Game\_Id%TYPE;

Customer\_Lname CUSTOMER.LNAME%TYPE;

Customer\_Fname CUSTOMER.FNAME%TYPE;

Game\_Bill NUMBER(10,2);

Bill NUMBER(10,2);

Price\_per\_hr GAME.Price\_Hr%TYPE;

Membership MEMBERSHIP\_DETAILS.MEMBERSHIP\_TYPE%TYPE;

Discount\_Amt MEMBERSHIP\_DETAILS.DISCOUNT%TYPE;

GameName GAME.Game\_Name%TYPE;

CURSOR REGISTER\_DETAILS IS

SELECT R.game\_id, R.total\_amount FROM REGISTERS R WHERE R.CUSTOMER\_ID = Customer\_Id AND R.BILL\_DATE = trunc(SYSDATE);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Bill');

DBMS\_OUTPUT.PUT\_LINE('==========================');

Bill := 0;

SELECT LNAME, FNAME into Customer\_Lname, Customer\_Fname FROM CUSTOMER C WHERE C.Customer\_Id = Customer\_Id;

SELECT DISCOUNT, MEMBERSHIP\_TYPE INTO Discount\_Amt, Membership FROM MEMBERSHIP\_DETAILS WHERE MEMBERSHIP\_ID = (SELECT MEMBERSHIP\_ID FROM CUSTOMER C WHERE C.CUSTOMER\_ID = Customer\_Id);

DBMS\_OUTPUT.PUT\_LINE('Customer Name: '||Customer\_Lname||', '||Customer\_Fname);

DBMS\_OUTPUT.PUT\_LINE('Game\_Id'||chr(9)||'Game\_Name'||chr(9)||'Hrs\_Played'||chr(9)||'Bill\_by\_Game');

DBMS\_OUTPUT.PUT\_LINE('=================================================================');

OPEN REGISTER\_DETAILS;

LOOP

FETCH REGISTER\_DETAILS INTO Gid, Game\_Bill;

EXIT WHEN REGISTER\_DETAILS%NOTFOUND;

SELECT G.Game\_Name, G.Price\_Hr into GameName, Price\_per\_hr FROM GAME G WHERE G.Game\_Id = Gid;

DBMS\_OUTPUT.PUT\_LINE(Gid||chr(9)||GameName||chr(9)||(Game\_Bill/Price\_per\_Hr)||chr(9)||Game\_Bill);

-- DBMS\_OUTPUT.PUT\_LINE(Game\_Id||' '||Game\_Bill);

Bill := Bill + Game\_Bill;

END LOOP;

CLOSE REGISTER\_DETAILS;

DBMS\_OUTPUT.PUT\_LINE('=================================================================');

DBMS\_OUTPUT.PUT\_LINE('Total Amount: '||'$'||Bill);

DBMS\_OUTPUT.PUT\_LINE('Membership: '||Membership);

Bill := Bill - (Bill \* (Discount\_Amt/100));

DBMS\_OUTPUT.PUT\_LINE('Discount %: '||Discount\_Amt);

DBMS\_OUTPUT.PUT\_LINE('Final Bill: '||'$'||Bill);

END GENERATE\_BILL; /

-- CALL GENERATE\_BILL(1);

Step 8: Rules by using Triggers

**1. Trigger to update the membership end date when the customer changes membership or takes a membership. Remove customer membership details when customer withdraws membership**

DROP TRIGGER END\_DATE\_TRIGGER; commit;

CREATE OR REPLACE TRIGGER END\_DATE\_TRIGGER

AFTER INSERT OR UPDATE OF MEMBERSHIP\_ID ON CUSTOMER

FOR EACH ROW

DECLARE

validity MEMBERSHIP\_DETAILS.VALIDITY%TYPE;

old\_date DATE;

end\_date DATE;

cid INT;

BEGIN

IF (:NEW.MEMBERSHIP\_ID > 0) THEN

-- DBMS\_OUTPUT.PUT\_LINE(:NEW.MEMBERSHIP\_ID);

cid := 0;

old\_date := :OLD.Membership\_Begin\_dt;

-- DBMS\_OUTPUT.PUT\_LINE(old\_date);

SELECT VALIDITY INTO validity FROM MEMBERSHIP\_DETAILS WHERE MEMBERSHIP\_ID = :NEW.MEMBERSHIP\_ID;

SELECT TO\_DATE(ADD\_MONTHS(:OLD.Membership\_Begin\_dt,validity)) into end\_date FROM DUAL;

SELECT count(\*) INTO cid FROM CUSTOMER\_MEMBERSHIP WHERE CUSTOMER\_ID = :OLD.CUSTOMER\_ID;

-- DBMS\_OUTPUT.PUT\_LINE(cid);

CASE WHEN cid > 0 THEN

UPDATE CUSTOMER\_MEMBERSHIP SET Membership\_End\_Dt = end\_date WHERE CUSTOMER\_ID = :OLD.CUSTOMER\_ID;

DBMS\_OUTPUT.PUT\_LINE('Membership details updated for '||:OLD.lname||chr(9)||', '||:OLD.fname);

ELSE

INSERT INTO CUSTOMER\_MEMBERSHIP (CUSTOMER\_ID, Membership\_End\_Dt) VALUES (:OLD.CUSTOMER\_ID, end\_date);

DBMS\_OUTPUT.PUT\_LINE('Membership details updated for '||:OLD.lname||chr(9)||', '||:OLD.fname);

END CASE;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Deleted Membership for the customer '||:OLD.lname||chr(9)||:OLD.fname);

DELETE FROM CUSTOMER\_MEMBERSHIP WHERE CUSTOMER\_ID = :OLD.CUSTOMER\_ID;

END IF;

END;

/

**2. On update of available units to zero of a game in a branch, trigger should make the game at that particular branch unavailable**

DROP TRIGGER GAME\_EQUIPMENT\_AVAILABILYT;

CREATE OR REPLACE TRIGGER GAME\_EQUIPMENT\_AVAILABILYT

AFTER UPDATE OF AVAILABLE\_UNITS ON GAME\_EQUIPMENT

FOR EACH ROW

DECLARE

AVA\_QNTY NUMBER;

B\_ID NUMBER;

E\_NAME VARCHAR(20);

G\_ID NUMBER;

BEGIN

CASE

WHEN UPDATING('AVAILABLE\_UNITS')

THEN

IF(:NEW.AVAILABLE\_UNITS = 0) THEN

B\_ID :=:NEW.BRANCH\_ID;

E\_NAME := :NEW.EQUIPMENT\_NAME;

SELECT S.GAME\_ID INTO G\_ID FROM EQUIP\_GAME S WHERE E\_NAME=S.EQUIPMENT\_NAME;

DELETE FROM HOSTS L WHERE L.BRANCH\_ID= B\_ID AND G\_ID=L.GAME\_ID;

END IF;

UPDATE GAME\_EQUIPMENT

SET AVAILABLE\_UNITS=0

WHERE BRANCH\_ID =1;