

## LAB 7:

**1. Product (BarCode, PName, Price, QuantityInStock)**

**Sale (SaleID, DeliveryAddress, CreditCard)**

**SaleItem (SaleID, BarCode, Quantity)**

**Create a trigger called updateAvailableQuantity that updates the quantity in stock in the Product table, for every product sold. The trigger should be executed after each insert operation on the SaleItem table: for the product with the given barcode (the one inserted into SaleItem), update the available quantity in Product table to be the old quantity minus the sold quantity.**

```
create table Product (  
    BarCode int,  
    PName varchar(40),  
    Price int,  
    QuantityInStock int,  
    primary key (BarCode));
```

```
create table Sale (  
    SaleID int,  
    DeliveryAddress varchar(40),  
    CreditCard int,  
    primary key(SaleID));
```

```
create table SaleItem (  
    SaleID int,  
    BarCode int,  
    Quantity int,  
    primary key (BarCode, SaleID),  
    foreign key (BarCode) references Product(BarCode));
```

```
create trigger updateavailabilityquantity
```

```
after insert on SaleItem for each row update Product set Product.Quantityinstock =  
Product.Quantityinstock - new.Quantity where new.Barcode = Product.Barcode;
```

```
insert into Product values (1, "chinnu", 5000, 20), (2, "ammu", 2000, 10);
```

```
insert into SaleItem values (1, 1, 2);
```

```

MySQL 8.0 Command Line Client
-> foreign key (BarCode) references Product(BarCode)
-> );
Query OK, 0 rows affected (0.39 sec)

mysql> create trigger updateavailabilityquantity
-> after insert on SaleItem for each row update Product set Product.QuantityInStock = Product.QuantityInStock - new.Quantity where new.Barcode = Product.Barcode;
Query OK, 0 rows affected (0.07 sec)

mysql> insert into SaleItem values (1, 4, 5);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('lab71`.`saleitem`, CONSTRAINT `saleitem_ibfk_1` FOREIGN KEY (`Barcode`) REFERENCES `product` (`barcode`))
mysql> insert into Product values (1, "chinnu", 5000, 20), (2, "ammu", 2000, 10);
Query OK, 2 rows affected (0.05 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> insert into SaleItem values (1, 1, 2);
Query OK, 1 row affected (0.06 sec)

mysql> select * from Product;
+-----+
| BarCode | PName | Price | QuantityInStock |
+-----+
| 1 | chinnu | 5000 | 18 |
| 2 | ammu | 2000 | 10 |
+-----+
2 rows in set (0.00 sec)

mysql> select * from SakaeItem;
ERROR 1146 (42S02): Table 'lab71.sakaeitem' doesn't exist
mysql> select * from SaleItem;
+-----+
| SaleId | BarCode | Quantity |
+-----+
| 1 | 1 | 2 |
+-----+
1 row in set (0.00 sec)

mysql> select * from Sale;
Empty set (0.00 sec)

mysql>

```

## 2. Activities (ActivityID, ActivityName)

Performers (PerformerID, PerformerName, Street, City, State, Zip, ActivityID)

Arenas (ArenaID, ArenaName, City, ArenaCapacity)

Concerts (PerformerID, ArenaID, ConcertDate, TicketPrice)

Create a trigger called deletePerformer that prevents the last performer for a particular activity to be deleted from the database. The trigger should be associated with a delete operation on the Performers table: if the performer to be deleted is the last one for his/her particular ActivityID then the performer cannot be deleted. Otherwise, the performer should be deleted.

create table Activites (

ActivityID int not null,

ActivityName varchar(40),

primary key (ActivityID)

);

create table Performers (

PerformerID int not null,

PerformerName varchar(40),

Street varchar(40),

```

    City varchar(40),
State varchar(40),
Zip int(6),
    ActivityID int,
    primary key (PerformerID),
foreign key (ActivityID) references Activites(ActivityID)
);

create table Arenas (
    ArenaID int not null,
    ArenaName varchar(40),
City varchar(40),
    ArenaCapacity varchar(40),
    primary key (ArenaID)
);

create table Concerts (
    PerformerID int,
    ArenaID int,
    ConcertDate datetime,
    TicketPrice int,
    primary key (PerformerID, ArenaID),
foreign key (PerformerID) references Performers(PerformerID),
    foreign key (ArenaID) references Arenas(ArenaID)
);
insert into Activites
values (1, "sports"), (2, "Swimming"),(3, "singing");

```

insert into Performers

values (1, "priyanka ", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 1),  
(2, "bharath", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 1),  
(3, "chinnu", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 2);

```
Select MySQL 8.0 Command Line Client
mysql> SELECT * FROM Performer;
-> DELETE FROM Performer WHERE PerformerID = 1;
-> select * from Performer;
-> //
-> END;
-> //
-> ^C
mysql> create table Employee (
-> Emp_no int,
-> Dept_no int,
-> Job varchar(20),
-> Salary int,
-> primary key(Emp_no),
-> foreign key (Dept_no) references Dept(Dept_no)
-> );
->
-> ^C
mysql> create table Employee (
-> Emp_no int,
-> Dept_no int,
-> Job varchar(20),
-> Salary int,
-> primary key(Emp_no),
-> foreign key (Dept_no) references Dept(Dept_no)
-> );
->
-> ^C
mysql> delimiter // CREATE TRIGGER deletePerformer BEFORE DELETE ON Performers FOR EACH ROW BEGIN IF (1 = (SELECT COUNT(PerformerID)
FROM Performers WHERE Performers.ActivityID = old.ActivityID ) IF (1 = (SELECT COUNT(PerformerID)
) THEN CALL RAISE_APPLICATION_ERROR('Deletion is Prevented!'); END IF; END; //
mysql> delete from Performers WHERE PerformerID=3;
-> delete from Performers WHERE PerformerID=1;
ERROR 1305 (42000): PROCEDURE lab72.RAISE_APPLICATION_ERROR does not exist
mysql> create trigger chkNull after insert on Dept for each row delete from Dept where new.Dept_no is null; insert into Dept values (null, 'Civil'
);
->
->
->
->
->
->
->
-> ^C
mysql> delimiter;
```

### 3. Employee(Emp\_no, Dept\_no, Job, Salary) Dept(Dept\_no, Dept\_name)

```
create table Employee (
    Emp_no int primary key,
    Dept_no int,
    Job varchar(20),
    Salary int
);
```

```
create table Dept (
    Dept_no int,
    Dept_name varchar(20)
);
```

insert into Dept  
values (1, 'CSE'),  
(2, 'ECE');

insert into Employee  
values (1, 1, 'professor', 50000),  
(2, 2, 'HOD', 100000);

**1. Write a TRIGGER to ensure that DEPT TABLE does not contain duplicate of null values in DEPTNO column.**

```
create trigger one
before insert
on Dept
for each row
begin
if ( new.Dept_no is null)
then
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "null not allowed";
end if;
end;
$$
```

**2. Write a Trigger to carry out the following action: on deleting a deptno from dept table , all the records with that deptno has to be deleted from the emp table**

```
create trigger onDlt
after delete
on Dept
for each row
delete from Employee where Employee.Dept_no = old.Dept_no;
```

```
delete from Dept where Dept_no=2;
```

```
select * from Employee;
```

The screenshot shows a MySQL 8.0 Command Line Client window with the following commands and output:

```
mysql> insert into Dept values (1, 'CSE'), (2, 'ECE');
Query OK, 2 rows affected (0.08 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> insert into Employee values (1, 1, 'professor', 50000), (2, 2, 'HOD', 100000);
Query OK, 2 rows affected (0.06 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from Dept;
+-----+
| Dept_no | Dept_name |
+-----+
| 1 | CSE |
| 2 | ECE |
+-----+
2 rows in set (0.10 sec)

mysql> select * from Employee;
+-----+
| Emp_no | Dept_no | Job | Salary |
+-----+
| 1 | 1 | professor | 50000 |
| 2 | 2 | HOD | 100000 |
+-----+
2 rows in set (0.00 sec)

mysql> create trigger chkNull after insert on Dept for each row delete from Dept where new.Dept_no is null; insert into Dept values (null, 'Civil');
Query OK, 0 rows affected (0.14 sec)

ERROR 1442 (HY000): Can't update table 'dept' in stored function/trigger because it is already used by statement which invoked this stored function/trigger.
mysql> create trigger onDlt after delete on Dept for each row delete from Employee where Employee.Dept_no = old.Dept_no;
Query OK, 0 rows affected (1.04 sec)

mysql> delete from Dept where Dept_no=2; select * from Employee;
Query OK, 1 row affected (0.06 sec)

+-----+
| Emp_no | Dept_no | Job | Salary |
+-----+
| 1 | 1 | professor | 50000 |
+-----+
1 row in set (0.00 sec)
```

A Zoom meeting overlay is visible in the bottom right corner with the title "New channel meeting 23:12" and a yellow circle containing the letters "KK". The Windows taskbar at the bottom shows the time as 17:14 on 19-10-2020.

**3. Write a trigger that raises an User defined error message and does not allow updating and insertion.**

```
create trigger insUpd
```

before insert

on Dept

for each row

CALL RAISE\_APPLICATION\_ERROR('auth error');

insert into Dept

values (3, 'EEE');