CS345 (Lab) LAB Assignment 3
Databases Laboratory
Triggers

Purpose: To learn how to create and use Database Triggers.

Note: This lab assignment is based on PL/SQL programming.

Introduction: A database trigger is a procedure that is attached to a table and executed in response to SQL INSERT, DELETE or UPDATE statements. The SQL statement, which causes the trigger to execute, is called a triggering event. When the trigger executes, we say that it fires. A trigger can fire before the triggering event or after the triggering event. It can fire once for each SQL statement (statement level trigger) or once for each row affected by the triggering event (row level trigger). A trigger is created using the create trigger statement which has the following syntax:

CREATE OR REPLACE TRIGGER trigger_name
{BEFORE | AFTER}
{DELETE | INSERT | UPDATE} [OF column_name]
ON table_name
[FOR EACH ROW]
[WHEN condition]
pl/sql block

Exercise 1: Using PL/SQL create an ITEM (Product, Quantity, Price, Cost) table in which Cost is a computed column. It stores cost of the item, which is computed as Price times Quantity.

Create ITEM table:

```
CREATE TABLE ITEM
(PRODUCT VARCHAR2(16),
QUANTITY NUMBER,
PRICE NUMBER,
COST NUMBER);
```

Insert the following row into table ITEM:

```
INSERT INTO ITEM(PRODUCT, QUANTITY, PRICE) VALUES ('Winter Tire', '4',' 90.00');
```

Now display the records from table ITEM using SELECT * FROM ITEM. You will see the values of cost column are missing from table.

Create a database trigger that update cost column automatically. Verify your trigger by following query.

INSERT INTO ITEM(PRODUCT, QUANTITY, PRICE) VALUES ('Winter Tire', '4',' 90.00');

Exercise 2: Create table GRADES with 5 columns:

ID	Number	(student id)
T1	Number	(mark from test 1)
T2	Number	(mark from test 2)
T3	Number	(mark from test 3)
MARK	Number	(average mark from test 1, 2 and 3)

Create a trigger MARK_TRG that calculates the value of the MARK column.

Assignment3: Create following tables

Product (productId, productName, categoryName, packageDesc, price)

```
CREATE TABLE Product (
productId int NOT NULL,
productName varchar(50),
categoryName varchar(50),
packageDesc varchar(50),
price decimal(9,2),
PRIMARY KEY (ProductId)
);
```

Customer (customerId, password, cname, street, city, state, zipcode, phone, email)

```
CREATE TABLE Customer (
customerId int NOT NULL PRIMARY KEY,
password VARCHAR(20) NOT NULL,
cname VARCHAR(50) NOT NULL,
street VARCHAR(50),
city VARCHAR(20),
state VARCHAR(2),
zipcode VARCHAR(10),
phone VARCHAR(10),
email VARCHAR(30) NOT NULL,
);
```

Orders (orderId, *customerId*, totalAmount)

Date: 9-Sep-2015

Triggers

```
CREATE TABLE Orders (
orderId int NOT NULL IDENTITY PRIMARY KEY,
customerId int,
totalAmount decimal(9,2)
CONSTRAINT FK_Orders_Customer FOREIGN KEY (customerId) REFERENCES
customer(customerId)
);
```

OrderedProduct(*orderId*, *productId*, quantity, price)

```
CREATE TABLE OrderedProduct (
orderId int NOT NULL,
productId int NOT NULL,
quantity int,
price decimal(9,2),
PRIMARY KEY (OrderId, ProductId),
CONSTRAINT FK_OrderedProduct_Order FOREIGN KEY (OrderId) REFERENCES
Orders(OrderId),
CONSTRAINT FK_OrderedProduct_Product FOREIGN KEY (ProductId) REFERENCES
Product (ProductId)
);
```

Assignment 3.1: Create a trigger that updates Orders.totalAmount whenever there is any change to the OrderedProduct table.

Assignment 3.2: Create a trigger that deletes all Orders and OrderedProduct records for a customer if that customer is deleted from the database.