Programming Lab 2

In this programming lab, we consider a Mail Order Database with seven tables.

ZIPCODES (zip, city)
EMPLOYEES (eno, ename, zip)
PARTS (pno, pname, qoh, price)
CUSTOMERS (cno, cname, street, zip, phone)
ORDERS (ono, cno, eno, received, shipped)
ODETAILS (ono, pno, qty)
ORDERS_ERRORS (Error_Date, ono, Error_Msg)

The **eno, pno, cno, ono** means the employee number, part number, customer number, and order number respectively. hdate in the EMPLOYEES table means the hiring date of an employee. **qoh** in the PARTS table means the quantity the part on hold. The **received** and **shipped** are the dates that the order is received and shipped. qty in the ODETAILS table means the quantity for the product for an order. ORDERS_ERRORS is a table to log order related error messages. Underlined attributes in each table are their corresponding primary keys.

Programming Tasks

Task1(15 points**)**: Create seven tables as described above. You should select proper primary key for each table. Foreign Keys shall also be defined to indicate the reference between tables. For example, **cno** and **eno** in the **ORDERS** table shall refer to **cno** in **CUSTOMERS** and **eno** in **EMPLOYEES**. Then, load data into these tables with scripts provided in 'Load-Data.sql'.

```
CREATE TABLE ZIPCODES (
    zip INT PRIMARY KEY,
    city VARCHAR(250)
);

CREATE TABLE EMPLOYEES (
    eno INT PRIMARY KEY,
    ename VARCHAR(255),
    zip INT,
    hdate DATE,
    FOREIGN KEY (zip) REFERENCES ZIPCODES(zip)
);

CREATE TABLE PARTS (
    pno INT PRIMARY KEY,
    pname VARCHAR(255) ,
    qoh INT,
    price INT
);
```

```
CREATE TABLE CUSTOMERS (
    cno INT PRIMARY KEY,
    cname VARCHAR(255),
    street VARCHAR(255),
    zip INT,
    phone VARCHAR(12)
    FOREIGN KEY (zip) REFERENCES ZIPCODES(zip)
);

CREATE TABLE ORDERS (
    ono INT PRIMARY KEY,
    cno INT,
    eno INT,
    received DATE,
    shipped DATE,
    FOREIGN KEY (cno) REFERENCES CUSTOMERS(cno),
    FOREIGN KEY (eno) REFERENCES EMPLOYEES(eno)
);

CREATE TABLE ODETAILS (
    ono INT,
    pno INT,
    qty INT,
    PRIMARY KEY(ono, pno),
    FOREIGN KEY (ono) REFERENCES ORDERS(ono),
    FOREIGN KEY (pno) REFERENCES PARTS(pno)
);

CREATE TABLE ORDERS_ERRORS (
    Error_Date DATE,
    ono INT,
    Error_Msg TEXT,
    FOREIGN KEY (ono) REFERENCES ORDERS(ono)
);
```

Task2(20 points**)**: In this task, you need to create a **trigger** to achieve: when an order is placed with the following two SQL scripts

INSERT INTO ORDERS values(500,3,1,'2017-3-18',null);

INSERT INTO ODETAILS values(500,10701,5);

The table PARTS shall have an automatic update for the quantity of the corresponding part ordered.

What to submit: The SQL Scripts of your Trigger. The Screenshot of **SELECT * FROM PARTS** after executing **INSERT INTO ODETAILS values(500,10701,5);**

```
delimiter ||
        CREATE TRIGGER newOrder AFTER INSERT ON ODetails
  FOR EACH ROW
  BEGIN
                UPDATE PARTS SET qoh = (qoh - new.qty) WHERE pno = new.pno;
        END;
```

||

	pno	pname	qoh	price
▶	10201	table	50	350
	10301	coffee table	30	150
	10401	tv stand	20	122
	10701	chair	45	51

O

Task3(25 points**)**: **Before working on this task, you must drop the trigger you created in task2.**

In Task2, no matter there is enough quantity on hold or not, the qoh in the PARTS table will be updated when an order is placed. In practice, it is possible that we do not have enough quantity for a new order. To avoid this kind of over ordering problem, the trigger shall only update the PARTS table when qoh is enough for the quantity in the new order; otherwise, an error message needs to be inserted in to the ORDERS_ERRORS table, which has today's date as Error_Date, "Quantity on hold is not enough!" as Error_Msg.

Note: To complete this task, please refer to the following example for how to include if-else statement in the trigger. We will have more detailed discussion in our follow up classes, but I encourage you to try your own practice first.

```
If (SELECT A FROM B WHERE C =1)>20 then
   SQL SCRIPTS;
ELSE
   SQL SCRIPTS;
END IF;
```

What to submit: The SQL Scripts for your Trigger. The Screenshot of **SELECT * FROM PARTS** and **SELECT * FROM ORDERS_ERRORS** after executing **INSERT INTO ODETAILS values(500, 10201,50);**

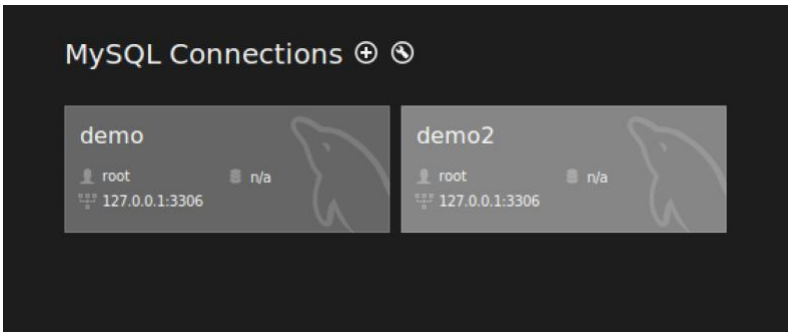
```
delimiter ||
        CREATE TRIGGER newError AFTER INSERT ON ODetails
  FOR EACH ROW
        BEGIN
        IF ((SELECT qoh FROM PARTS WHERE pno = new.pno) > 20) then
                UPDATE PARTS SET qoh = (qoh - new.qty) WHERE pno = new.pno;
    ELSE
                INSERT INTO ORDERS_ERRORS VALUES (curdate(), 'Quantity on hold is not enough!');
    END IF;
        END;
||
```

pno	pname	qoh	price
10201	table	0	350
10301	coffee table	30	150
10401	tv stand	20	122
10701	chair	45	51

Error_Date	ono	Error_Msg

Task4(40 points):

Part A: In this task, you need to open two sessions and set their **session isolation_level** as **SERIALIZABLE**. To create two sessions at the same time, you can create two connections in MySQL Workbench as follows



You can also create one connection in MySQL Workbench and the other one using terminal.

Then, you need to perform the following steps:

1. In Session 1, start a transaction and insert yourself as a customer into the CUSTOMERS table. Do not commit at this moment.
2. In Session 2, query the CUSTOMERS table. What is your query results?
3. In Session 1, commit your transaction.
4. In Session 2, query the CUSTOMERS table again. What is your query results?

Repeat the above steps again for **isolation_level** at **READ UNCOMMITTED.**

What to submit: Your observation for each step , and explain your observation.

After insert:

In session 2, the customers table is updated

After commit:

In session 2, the customers table is updated

Part B: Create two connection sessions and set their session **isolation_level** at **SERIALIZABLE.** Then, you need to perform the following steps:

1. In Session 1, start a transaction and update the name of customer with cno = 3 to ‘C3’. Do not commit at this moment.
2. In Session 2, start a transaction and try to update the name of customer with cno = 3 to ‘C32’. What is your observation?

3. In Session 1, rollback your transaction. What is your observation in Session 2?
4. In Session 2, rollback your transaction.

What to submit: Your observation for each step, and explain your observation.

After Insert:

In Session 2, the customer table is not updated

After rollback

In Session 2, customer table is updated