**Jack Ryan**

**COEN 178 Intro to Database Systems Spring 2019**

**Lab 6 (100 pts)**

**Objectives: Learn**

* To practice writing triggers to enforce integrity constraints between tables.
* Using PHP and Oracle SQL to build a Web application. PHP generates the HTML forms necessary to get the user input, accesses the database and generates the output.

**Submit:** A SQL file with your modified trigger from exercise 1, and the completed triggers from 2 & 4. Also submit a text file with the answers to the questions in 2-5.

Question 6 should be completed and submitted to the TA before you start Lab 7.

**Demo:** The webpage in part 2.

**Part 1**

In this part, you will practice writing a few triggers to enforce business rules among the table data.

**Create the following tables for Bank Database.**

**Create table BANKCUST\_6 (custno VARCHAR(5) Primary Key,custname VARCHAR(20),street VARCHAR(30), city VARCHAR(20));**

**Create table ACCOUNTS\_6 (AccountNo VARCHAR(5) Primary Key,accountType VARCHAR(10), amount NUMBER(10,2), custno varchar(5),**

**CONSTRAINT accounts\_fkey FOREIGN Key (custno)REFERENCES BANKCUST\_6(custno));**

**Create table TOTALS\_6 (custno VARCHAR(5), totalAmount Number(10,2), CONSTRAINT totals\_fkey FOREIGN Key (custno)REFERENCES BANKCUST\_6(custno));**

**Exercise 1 (10 pts)**

In this exercise, you will write a trigger display the data that is inserted into **Bankcust\_6** table. This trigger is not really very useful, but just a warm up exercise to write a trigger and see if it fires correctly, as a preparation to write the triggers in the subsequent exercises.

1. **At SQL PROMPT type set serveroutput on;**
2. **Create the following trigger (either run from a text file or copy and paste it at SQL prompt)**

**CREATE or REPLACE TRIGGER display\_customer\_trig**

**AFTER INSERT on BankCust\_6**

**FOR EACH ROW**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('From Trigger '||'Customer NO: '||:new.custno||' Customer Name: '||:new.custname);**

**END;**

**/**

**show errors;**

1. Insert the following values into BANKCUST\_6 table.

**insert into BANKCUST\_6 values('c1','Smith','32 Lincoln st','SJ');**

**insert into BANKCUST\_6 values('c2','Jones','44 Benton st','SJ');**

**insert into BANKCUST\_6 values('c3','Peters','12 palm st','SFO');**

**insert into BANKCUST\_6 values('c20','Chen','20 san felipo','LA');**

**insert into BANKCUST\_6 values('c33','Williams',' 11 cherry Ave','SFO');**

**Did your trigger work?**

**Yes the trigger executed successfully.**

**Modify the trigger so that it displays the city as well.**

**Added a statement to print :new.city**

**Exercise 2 (15 pts)**

1. We will disable the **display\_customer\_trig** using the alter trigger statement. Use the statement (use the trigger name).

**Alter trigger *trigger\_name* disable**

1. We will now write a trigger which fires after inserting a row in the Accounts\_6 table. The trigger should enter the custno and the amount into the TOTALS\_6 table as follows:

* If the custno is already in Totals\_6 table, adds the new amount to the existing one.
* If the custno is not in Totals\_6 table, adds a new row for this new customer.

The Totals\_6 table should give us the total amount in all the accounts owned by each customer.

1. Complete the code for the trigger given below, following the comments

**Create Or Replace Trigger Acct\_Cust\_Trig**

**AFTER INSERT ON Accounts\_6**

**FOR EACH ROW**

**BEGIN**

**/\*If the custno is already in the Totals\_6 table, the update will succeed \*/**

**update totals\_6**

**set totalAmount = totalAmount + :new.amount**

**where custno = :new.custno;**

**/\*If the custno is not in the Totals\_6 table, we insert a row into**

**Totals\_6 table. Complete the missing part in the subquery \*/**

**insert into totals\_6 (select :new.custno, :new.amount from dual**

**where not exists (select \* from TOTALS\_6 where custno= ));**

**END;**

**/**

**Make sure that your trigger compiles without any errors.**

1. **Delete if there is any data in the Accounts\_6 and Totals\_6 tables.**
2. **Insert the following data into Accounts\_6 table.**

insert into ACCOUNTS\_6 values('a1523','checking',2000.00,'c1');

insert into ACCOUNTS\_6 values('a2134','saving',5000.00,'c1');

insert into ACCOUNTS\_6 values('a4378','checking',1000.00,'c2');

insert into ACCOUNTS\_6 values('a5363','saving',8000.00,'c2');

insert into ACCOUNTS\_6 values('a7236','checking',500.00,'c33');

insert into ACCOUNTS\_6 values('a8577','checking',150.00,'c20');

Did your trigger work? How did you check?

**Yes the trigger works. I tested it by printing both tables after the inserts and comparing the Amount values for each custno.**

Show the data in Totals\_6 table.

**CUSTN TOTALAMOUNT**

**----- -----------**

**c1 7000**

**c2 9000**

**c33 500**

**c20 150**

What is the amount for the customer, ‘c1’?

**$7000.00**

Does the total amount for ‘c1’ agree with the amounts for that customer in the Accounts\_6 table?

**Yes it agrees correctly. 2 inserts, $2000.00 + $5000.00 = $7000.00**

**Exercise 3 (15 pts)**

If your trigger is working correctly and updating the total amount for a customer every time a new Account for that customer is created, let us try the following query from SQL prompt.

**update Accounts\_6**

**set amount = 1000**

**where accountno = 'a1523';**

**If the above query successfully ran, check the Totals\_6 table.**

**CUSTN TOTALAMOUNT**

**----- -----------**

**c1 7000**

**c2 9000**

**c33 500**

**c20 150**

**What is the amount for the customer, ‘c1’?**

**$7000.00**

**Does the amount in Totals\_6 table for ‘c1’ agree with the total of amounts in all the accounts for ‘c1’ in Accounts\_6 table?**

**No, it does not match. The trigger did not correctly update the total on subtraction of assets.**

**Exercise 4 (15 pts)**

We will modify our trigger **Acct\_Cust\_Trig** to fire after inserting as well as updating data in the Accounts\_6 table.

Create Or Replace Trigger **Acct\_Cust\_Trig**

AFTER INSERT OR UPDATE ON Accounts\_6

FOR EACH ROW

BEGIN

If **inserting** then

update totals\_6

set totalAmount = totalAmount + :new.amount

where custno = :new.custno;

insert into totals\_6 (select :new.custno, :new.amount from dual

where not exists ( **write your complete query from Question 2**);

END IF;

if **updating** then

**/\* If we are updating we want to correctly set the totalAmount**

**to the new amount that may be >= or < old amount**

**Complete the query \*/**

update totals\_6

set totalAmount = totalAmount +

where custno = :new.custno;

end if;

END;

/

Show Errors;

1. Complete and Compile your trigger.
2. Now delete all rows from Accounts\_6 table and Totals\_6 table.
3. **Insert the following data into Accounts\_6 table.**

insert into ACCOUNTS\_6 values('a1523','checking',2000.00,'c1');

insert into ACCOUNTS\_6 values('a2134','saving',5000.00,'c1');

insert into ACCOUNTS\_6 values('a4378','checking',1000.00,'c2');

insert into ACCOUNTS\_6 values('a5363','saving',8000.00,'c2');

insert into ACCOUNTS\_6 values('a7236','checking',500.00,'c33');

insert into ACCOUNTS\_6 values('a8577','checking',150.00,'c20');

1. Show the data in Totals\_6 table.

**CUSTN TOTALAMOUNT**

**----- -----------**

**c1 7000**

**c2 9000**

**c33 500**

**c20 150**

What is the amount for the customer, ‘c1’?

**$7000.00**

1. Run this query

**update Accounts\_6**

**set amount = 1000**

**where accountno = 'a1523';**

1. **If the above query successfully ran, check the Totals\_6 table.**

**CUSTN TOTALAMOUNT**

**----- -----------**

**c1 6000**

**c2 9000**

**c33 500**

**c20 150**

1. **What is the amount for the customer, ‘c1’?**

**$6000.00**

1. **Does the amount in Totals\_6 table for ‘c1’ agree with the total of amounts in all the accounts for ‘c1’ in Accounts\_6 table?**

**Yes, it agrees. $5000.00 + $1000.00 = $6000.00**

**Exercise 5 (10 pts)**

One way to check if a specific column is being updated in a table is to use, if updating (column name).

The following trigger prevents the primary key in the BANKCUST\_6 table from being updated.

**Create Or Replace Trigger NoUpdatePK\_trig**

**After UPDATE ON BANKCUST\_6**

**For each row**

**BEGIN**

**if updating ('custno') then**

**raise\_application\_error (-20999,'Cannot update a Primary Key');**

**End if;**

**END;**

**/**

**show errors;**

Now, type the following command from SQL prompt:

**UPDATE BANKCUST\_6**

**Set custno=’c99’**

**Where custno=’c1’;**

**What is the result?**

**ERROR at line 1:**

**ORA-20999: Cannot update a Primary Key**

**ORA-06512: at "JRYAN.NOUPDATEPK\_TRIG", line 3**

**ORA-04088: error during execution of trigger 'JRYAN.NOUPDATEPK\_TRIG'**

**Is the custno updated?**

**No, the customer number for ‘c1’ is still ‘c1’. Update was prevented.**

**--------------------------------------------------------------------------------------------------------------------------------**

**Question 6 (20 pts)**

Question 6 should be completed and submitted to the TA before you start Lab 7.

In this question, you will create tables, **M\_Student, M\_Course, M\_CoursesTaken, M\_PrereqCourse and M\_CourseRegister.** You will populate the tables with the given test data.

You will write a **trigger** that will check, when a student registers for a course, if the student has taken the *prereqs* for that course. If so, the student no and course no. are entered in M\_CoursesTaken table. If not, the registration will fail.

1. Create the following tables:

/\* A table to hold student data \*/

**Create table M\_Student (stno varchar(3) primary key, name varchar(20));**

/\* A table to hold course info.\*/

**Create table M\_Course (courseno varchar(5) primary key, units Integer);**

/\* A table to hold the info of courses taken by a student \*/

**Create table M\_CoursesTaken (stno varchar(3),courseno varchar(5));**

/\* A table to hold registration of a student into a course.\*/

**Create table M\_CourseRegister(stno varchar(3),courseno varchar(5));**

/\* A table to hold course and its prerequisites.\*/

**Create table M\_PreReqCourse (courseno varchar(5), prereq varchar(5));**

1. Complete the trigger below using the comments given

CREATE or Replace TRIGGER **CheckPrereq\_Trig**

**AFTER insert** ON **M\_CourseRegister**

FOR EACH ROW

DECLARE

**/\* local variables you will need \*/**

l\_cnt Integer;

l\_no Integer;

BEGIN

**/\* Check if the course has a prereq.in M\_PrereqCourse table**

**If it does not have a prerequisite, it will**

**Have a null for prereq. Complete this query**

**\*/**

Select Count(prereq) into l\_no

from M\_PrereqCourse

WHERE CourseNo = :NEW.courseno

And

**/\* prereq not null. \*/**

**/\* If there are prereqs \*/**

IF l\_no > 0 THEN

Select Count(prereq) into l\_cnt

from M\_PrereqCourse

WHERE CourseNo=:NEW.courseno

AND

prereq NOT IN (

**/\* Select course no from**

**M\_CoursesTaken where stNo = :NEW.stNo**

**Complete this query \*/**

if l\_cnt > 0 THEN

**/\* There are prereqs not taken by the student\*/**

DBMS\_OUTPUT.PUT\_LINE ('No prereq');

RAISE\_APPLICATION\_ERROR(-20010,'prereqs not done');

END IF;

END IF;

**/\* If prereqs have been taken by the student, registration**

**Is successful and the new course is entered into M\_CoursesTaken table.**

**\*/**

insert into M\_CoursesTaken values (:NEW.stNo,:NEW.CourseNo);

END CheckPrereq\_Trig;

/

Show Errors;

C) Compile your Trigger.

1. **Enter the following data into the tables.**

**/\* Student data \*/**

insert into M\_Student values ('s1','smith');

insert into M\_Student values ('s2','jones');

**/\* Course data \*/**

insert into M\_course values ('c1',4);

insert into M\_Course values ('c2',2);

insert into M\_course values ('c3',4);

insert into M\_Course values ('c4',2);

**/\* Prerequisite data \*/**

insert into M\_PrereqCourse values('c3','c1');

insert into M\_PrereqCourse values('c4','c1');

insert into M\_PrereqCourse values('c4','c2');

insert into M\_PrereqCourse values('c1',NULL);

insert into M\_PrereqCourse values('c2',NULL);

1. Now is the time to check if your trigger is firing correctly.

**Enter the following data into the tables.**

/\* This should succeed, since c1 does not have any prereqs \*/

insert into M\_CourseRegister values ('s1','c1');

/\* This should succeed, since c3 has c1 as prereq, which s1 has taken. \*/

insert into M\_courseRegister values('s1','c3');

**/\* Do a Select \* from M\_CoursesTaken \*/**

/\* This should fail and fire the trigger, since c4 has c1 and c2 as prereqs

And s1 has taken only one. \*/

insert into M\_courseRegister values('s1','c4');

**/\* Do a Select \* from M\_CoursesTaken. This**

**Should show only two courses(c1 and c3) for s1 so far. \*/**

/\* This should succeed, since c2 does not have any prereqs \*/

insert into M\_courseRegister values('s1','c2');

/\* This should succeed now, since s1 has completed the two prereqs \*/

insert into M\_courseRegister values('s1','c4');

**/\* Do a Select \* from M\_CoursesTaken \*/**

/\* Will this fail ? \*/

insert into M\_courseRegister values('s2','c4');

**It will fail because ‘s2’ has not completed the prereq ‘c2’**

**----------------------------------------------------------------------------------------------------------------------------------**

**Part 2 (15 pts)**

In this, you will run a PHP program that will create an HTML form to get user’s input, connect to your Oracle database tables, fetches and shows the data as given in the query.

**Use the file, showSalary\_form.php, place the file in the folder where you are required to put the .php files.**

**You must edit the code below to put your login and password. At the end of the lab session, please feel free to change your password.**

1. **Run the program by typing the URL of the program in the browser window. Give an employee name that you have stored in your AlphaCoEmp table. Check if the program works.**
2. **Now, change the code in the showSalary\_form.php file, to display the name, salary and title. Try not to display this information from the function (getSalaryFromDB) but from the main program that calls this function.**