**CSCI 4125/5125 Course Project**

**Data Models and Database Systems**

**Fall 2023**

**Course Project**

**Phase 8: PL/SQL**

**Due: Sunday, 11/12 @ 11:59pm**

**Reading:** Silberschatz Chapter 5

**Submission Guidelines:**

1. This assignment is worth 70 points for all students.

2. It is your responsibility to make sure all files are readable and submitted on time.

**Submission:**

* Task 1: Submit an .sql file containing your trigger for a total of 30 points.
* Task 2: Submit an .sql file containing your procedure for a total of 40 points.

Note that the 70 points for this project phase is more than the original 55 points proposed in the project description. This additional 15 points is essentially extra credit.

**Task1. PL/SQL Triggers (30 points)**

**Introduction.** Your task is to write a trigger that will maintain the inventory for products. While you are not required to use DBMS\_OUTPUT.PUT\_LINE, I highly recommend using it for testing and debugging your code. Note that this is an example of maintaining a computed column that we saw in Chapter 6: E-R Modeling. First, add a column to the Product table to store this value using the following command:

ALTER TABLE Product ADD Inventory NUMBER DEFAULT 64;

If you make a mistake along the way, you can reset this column with a simple update:

UPDATE Product SET Inventory = 64;

**Functionality.** Write a trigger that will fire when you INSERT a row into the Lineitem table. This trigger will check the value of Inventory for the corresponding ProductID.

* If Inventory is currently 0, then the product is out of stock. In this case, return an error message “Product #[insert the ProductID] is out of stock!” and cancel the insert.
* If the Inventory minus the quantity is less than 0, then we cannot fulfill this order. In this case, return an error message “Product #[insert the ProductID] does not have enough stock for this order!” and cancel the insert.
* If the Inventory minus the quantity is greater than or equal to 0, we can fulfill the order. In this case, update the product inventory to reflect this order.

**Testing.** You can test your trigger with the following:

* Delete all records from Lineitem: DELETE FROM Lineitem;
* Run your .sql script containing your INSERT statement for the Lineitem table.
* Three inserts should generate error messages that look similar to those below:

For Order O0011, P05:

INSERT INTO lineitem VALUES('O0011', 'P05', 200)

Error at Command Line : 68 Column : 13

Error report -

SQL Error: ORA-20202: Product #P05 does not have enough stock for this order!

ORA-06512: at "JAYSTUDENT.QUANTITYUPDATE", line 11

ORA-04088: error during execution of trigger 'JAYSTUDENT.QUANTITYUPDATE'

Order O0041, P20:

INSERT INTO lineitem VALUES('O0041', 'P20', 75)

Error at Command Line : 122 Column : 13

Error report -

SQL Error: ORA-20202: Product #P20 does not have enough stock for this order!

ORA-06512: at "JAYSTUDENT.QUANTITYUPDATE", line 11

ORA-04088: error during execution of trigger 'JAYSTUDENT.QUANTITYUPDATE'

Order O0045, P18

INSERT INTO lineitem VALUES('O0045', 'P18', 5)

Error at Command Line : 127 Column : 13

Error report -

SQL Error: ORA-20201: Product #P18 is out of stock!

ORA-06512: at "JAYSTUDENT.QUANTITYUPDATE", line 9

ORA-04088: error during execution of trigger 'JAYSTUDENT.QUANTITYUPDATE'

* The query SELECT ID, Inventory FROM Product ORDER BY ID; should return the following output:

ID INVENTORY

--- ----------

P01 39

P02 27

P03 30

P04 51

P05 37

P06 54

P07 30

P08 53

P09 44

P10 47

P11 39

P12 54

P13 54

P14 49

P15 39

P16 28

P17 39

P18 0

P19 16

P20 33

* The Lineitem table should have 97 records since three of the insert were rejected.

*Hint:* You should be able to write this trigger in ~30 lines of code. If your solution uses more than 50 lines, I encourage you to ask questions in class or office hours.

**Task 2. PL/SQL Procedures (40 points)**

**Introduction.** Your task is to write a PL/SQL procedure that generates an invoice for a given customer ID and order date. The invoice will compute some of the derived columns we saw in Phase 2, specifically product cost, shipping cost, and total cost. You will use DBMS\_OUTPUT.PUT\_LINE to print your invoice text. Keep in mind that it may be easier to start by building an anonymous PL/SQL that you can later store as a procedure.

**Functionality.** Your procedure must include the following:

* Accept two arguments: a customer ID and an order date.
* If the customer has not placed an order on the given date, print a message “No invoice to generate” and return.
* Provide an invoice header that includes: order date, customer ID, and customer name.
* List all products in the order, the quantity, the price/unit, and the total line price (i.e., quantity \* price/unit).
* Display the total cost for all products, the shipping cost ($0 if total product cost >= $35, else $10), and the total cost (i.e., total product cost plus shipping cost).
* You are welcome to add additional information if you’d like to.

**Testing.** The next page contains some example procedure calls along with the invoice they return. Your formatting does not need to match mine exactly, but it must contain all of the relevant information listed above.

*Hint:* You will need to use a cursor, but not more than one. If you find yourself writing several cursors, 1) look at how you could improve your cursor and 2) ask if you could solve the problem with a SELECT INTO.

*Hint:* You should be able to write this procedure in ~50 - 60 lines of code. If your solution uses more than 80 lines, I encourage you to ask questions in class or office hours.

*Hint:* If you want to format your output similar to mine, PL/SQL supports the LPAD() and RPAD() functions you have probably seen in other programming languages. You can also include decimal places with TO\_CHAR(). Here is an example of this that you can run:

SELECT LPAD('$' || TO\_CHAR(5, '9999.99'), 10) AS "Cost" FROM Dual;

A screenshot of a computer

Description automatically generated