



Tecnológico de Monterrey

1st partial Evaluation Last Homework

Students:

Salvador Orozco Villalever - A07104218

Aranzza Abascal Fararoni - A01329203

Professor:

Dr. Daniel Pérez Rojas

Subject:

Advanced Databases

Due date:

February 19th, 2018

ITESM Campus Puebla

1. Transaction

Sales operations were implemented as transactions to avoid products being taken off from the stock without confirming the payment, or in case the system crashes, the Internet connection is lost, etc..

```
buy.php — DepartmentStoreSystem

53 $link->begin_transaction(MYSQLI_TRANS_START_READ_WRITE);
54
55 // Mysql query to insert a new customer depending on user input
56 $sql_insertCustomer = "INSERT INTO customer (id,name,last_name) values(null,'" . "$name" . "','" . "$lastname" . "')";
57 $result_insertCustomer = $link->query($sql_insertCustomer);
58
59 // Mysql query to get the last insert id in customer table
60 $sql_getRecentlyInsertedCustomerID = "SELECT LAST_INSERT_ID() INTO @newCustomer_id";
61 $link->query($sql_getRecentlyInsertedCustomerID);
62
63 // Mysql query to insert a new sale, using the last insert id
64 $sql_createSale = "INSERT INTO sale (id,customer,date_time) values(null,@newCustomer_id,NOW())";
65 $result_insertSale = $link->query($sql_createSale);
66
67 // Mysql query to get the last insert id in sale table
68 $sql_getRecentlyInsertedSaleID = "SELECT LAST_INSERT_ID() INTO @newSale_id";
69 $link->query($sql_getRecentlyInsertedSaleID);
70
71 // Variables to validate if the following queries are correct
72 $allSaleProductsInsertionsAreOk = true;
73 $allSKUUpdatesAreOk = true;
74
75 // Traverse through the map of (key=product_id, value=product_amount)
76 foreach($query_array as $product_id => $product_amount) {
77
78     // Mysql query to insert a new sale_product
79     $sql_createSaleProduct = "INSERT INTO sale_product (sale,product,quantity) values(@newSale_id,'" . $product_id . "','" . $product_amount)";
80     $result_insertSaleProduct = $link->query($sql_createSaleProduct);
81
82     // Mysql query to get stock of the product the user will buy
83     $sql_selectCurrentSKU = "SELECT sku FROM product WHERE id = " . $product_id;
84     $result_selectedSKU = $link->query($sql_selectCurrentSKU);
85
86     // While loop to store the query value in a php variable and subtract the number of products the user will buy: generating new stock value
87     while($row = $result_selectedSKU->fetch_assoc()){
88         $result = $row['sku'] - $product_amount;
89     }
90
91     // Validation: sku cannot be negative, it cannot sell more products than stock number
92     if($result >= 0){
93         // Mysql query to update stock value
94         $sql_updateSKU = "UPDATE product SET sku = $result WHERE id = " . $product_id;
95         $result_updateSKU = $link->query($sql_updateSKU);
96     } else {
97         // Redirect to error page
98         header("Location: error.php");
99         die;
100     }
101
102     // Flags to know if transactions where correctly done
103     if($result_insertSaleProduct == false AND $result_updateSKU == false AND $result_selectedSKU == false){
104
105         $allSaleProductsInsertionsAreOk = false;
106         $allSKUUpdatesAreOk = false;
107         break;
108     }
109 }
110 // Validation of flags to commit
111 if ($result_insertCustomer and $result_insertSale and $result_selectedSKU and $allSaleProductsInsertionsAreOk and $allSKUUpdatesAreOk) {
112     $link->commit();
113 } else {
114     $link->rollback();
115 }
```

Figure 1: *PHP transaction for the sales*

2. Event

An event was created such that every hour it stores the amount of sales in the last hour (starting at 5:00 hrs and ending at 23:00 hrs). It includes a transaction to guarantee the database's integrity.

```
DELIMITER //
```

```
CREATE EVENT sales_per_hour_event ON SCHEDULE EVERY 1 HOUR  
ON COMPLETION PRESERVE  
DO  
IF CURRENT_TIME() >= '06:00:00' AND CURRENT_TIME() <= '23:00:00' THEN  
  
START TRANSACTION;  
SELECT @sales_amount := compute_sales_amount(NOW() - INTERVAL 1 HOUR, NOW() );  
INSERT INTO sales_per_hour VALUES(NULL, NOW() - INTERVAL 1 HOUR, NOW(), @sales_amount);  
COMMIT;  
END IF//  
DELIMITER ;
```

3. Trigger

A trigger was implemented such that when the stock of a product reaches 0 units, it generates a purchase order to get n more products, where n corresponds to the minimum sku value of that specific product.

```
DELIMITER $$  
CREATE TRIGGER after_product_sale_sku0  
  AFTER UPDATE ON product  
  FOR EACH ROW  
  BEGIN  
  
    DECLARE purchase_order_id INT DEFAULT 0;  
  
    IF NEW.sku = 0 THEN  
  
      INSERT INTO purchase_order VALUES (  
        NULL, NULL, NOW(), NULL, "SUPPLIER01234567"  
      );  
  
      SET purchase_order_id := LAST_INSERT_ID();  
  
      INSERT INTO purchase_order_product(  
        purchase_order, product, quantity, price  
      ) VALUES(  
        purchase_order_id, OLD.id, OLD.mi_req_sku, OLD.price  
      );  
  
    END IF;  
  
  END  
$$
```

```
DELIMITER ;
```

4. Stored procedure

A stored procedure was created for the sales part. It is called from PHP.

```
DELIMITER //
CREATE PROCEDURE GetCategoryProduct (IN categoryID VARCHAR(10))
    BEGIN
        SELECT p.id, p.name, p.sku, d.name
        FROM product p
        JOIN category c
        ON p.category = c.id
        JOIN department d
        ON c.department = d.id
        JOIN branch b
        ON d.branch = "B0710"
        WHERE c.id = categoryID
        GROUP BY p.id
        ORDER BY d.name;
    END //
DELIMITER ;
```

5. Function

A function was implemented to compute the sum of the amounts of all sales on a given date.

```
DELIMITER $$
CREATE FUNCTION compute_sales_details_on_day (date_param DATE)
RETURNS DOUBLE
DETERMINISTIC
BEGIN
    DECLARE sales_amount_on_day DOUBLE;
    SET sales_amount_on_day = (SELECT SUM(daySalesQuery.sale_total_amount)
    FROM
        (SELECT SUM(sp.quantity*product.price) as sale_total_amount
        FROM sale_product sp
        LEFT JOIN product ON sp.product = product.id
        LEFT JOIN sale s ON sp.sale = s.id
        LEFT JOIN customer c ON s.customer = c.id
        WHERE DATE(s.date_time) = date_param
        GROUP BY sp.sale) AS daySalesQuery);

    RETURN sales_amount_on_day;
END $$
DELIMITER ;
```

6. Web interface



Figura 2: Home page based on one page bootstrap template. The menu is accessed using tabs in the upper-right section.



Figura 3: Buy section on home page to select category.

LIVERPOOL

COMPRAR

PROVEEDORES

INVENTARIO

VENTAS

ORDENES DE COMPRA

COMPRAR

Categoría

Ver artículos

NOMBRE PRODUCTO	DEPARTAMENTO	PRECIO UNITARIO	CANTIDAD
PS4 WRC 7	Videojuegos	\$1149	0
PS4 Shadow of the Colossus	Videojuegos	\$1099	0
Xbox One Metal Gear Survive	Videojuegos	\$899	0

Compra

Figura 4: Once the user has selected a category, products are shown in a table.

LIVERPOOL

COMPRAR

PROVEEDORES

INVENTARIO

VENTAS

ORDENES DE COMPRA

COMPRA A PROVEEDORES

NOMBRE PRODUCTO	DEPARTAMENTO	PRECIO UNITARIO	CANTIDAD
Playera cuello redondo	ÉI	\$399	0
Pulsera Lombrozo gris acero	ÉI	\$399	0
Camisa casual a rayas	ÉI	\$599	0
Tenis de piel	ÉI	\$3299	0
Chamarra lisa	ÉI	\$2999	0
Pantufla Polo Ralph Rauren	ÉI	\$699	0
Platillo Zildjian	Electrónicos	\$4029.35	0

Figura 5: Section to create a purchase order to supplier, which increments stock products.

INVENTARIO

★

ID	NOMBRE PRODUCTO	DEPARTAMENTO	PRECIO UNITARIO	STOCK
1058492732	Playera cuello redondo	Él	\$399	45
1035829520	Pulsera Lombrozo gris acero	Él	\$399	360
1048392058	Camisa casual a rayas	Él	\$599	56
1054832049	Tenis de piel	Él	\$3299	89
1058492302	Chamarra lisa	Él	\$2999	43
1058493741	Pantufia Polo Ralph Rauren	Él	\$699	120
1035139481	Platillo Zildjian	Electrónicos	\$4029.35	68
1034581031	30 de Febrero Ha-Ash	Electrónicos	\$135.2	140

Figura 6: Stock section which shows stock number for each product.

LIVERPOOL

COMPRAR

PROVEEDORES

INVENTARIO

VENTAS

ORDENES DE COMPRA

VENTAS

★

ID	Fecha y hora	Nombre del Cliente	Importe total
1	2018-02-02 00:00:00	Juan Gonzalez	\$113713.4
2	2018-02-02 00:00:00	Julio Regalado	\$12749.2
13	2018-02-13 01:31:53	Dan Perez	\$2793
15	2018-02-13 01:33:29	Juanito Banana	\$167880
16	2018-02-13 01:34:42	Ciro Gomez	\$899
17	2018-02-18 13:59:11	Dan Gas	\$1596
18	2018-02-18 18:30:46	klj klj	\$798
19	2018-02-18 18:52:25	Salvador Orozco	\$21945

Figura 7: Sales section which shows all the sales have been done.

ORDENES DE COMPRA

★

ID	Fecha y hora	Proveedor	Producto	Cantidad	Importe total
1	2018-02-18 18:48:33	SUPPLIER01234567	Playera cuello redondo	2	\$ 798
2	2018-02-18 18:48:40	SUPPLIER01234567	Playera cuello redondo	2	\$ 798
3	2018-02-18 18:49:01	SUPPLIER01234567	Playera cuello redondo	2	\$ 798
4	2018-02-18 18:49:13	SUPPLIER01234567	Playera cuello redondo	2	\$ 798
5	2018-02-18 18:53:57	SUPPLIER01234567	Playera cuello redondo	50	\$ 19950

Figura 8: Section which shows all the purchase orders the company has done.

LIVERPOOL

INFORMACIÓN DEL CLIENTE

★

NOMBRE:

Dan

APELLIDO:

Pérez

PRODUCTOS SELECCIONADOS

★

ID	Nombre	Importe
1032582351	Tenis lisos	\$ 6897
MONTO TOTAL DE LA COMPRA		\$ 6897

Figura 9: Redirection to confirmation page to get the customer information and insert it into the table.

COMPRA NO REALIZADA



[Regresar a la tienda.](#)

Figura 10: *Redirection to error page when the sale transaction has to rollback.*