CSE

4126

DISTRIBUTED DATABASE MANAGEMENT SYSTEM

PROJECT REPORT

GO Cart

SUBMITTED BY

SAYED HOSSAIN KHAN | 15-01-04-133

GROUP MEMBERS

AREFEEN SULTAN | 15-01-04-111

TONMOY SINHA | 15-01-04-129

Abstract Of Project:

People visits super shop for buying the best products. But they need a proper guidance for which product is more reliable and usable for them. For handling it swiftly we need a management system. Our target is to make shopping much more reliable and serviceable. For managing super shop, we are using distributed database management system. Here's some of the services are given below:

1. Transaction between Customer and Local Store.
2. Membership Approval for customer.
3. Customer can search item for information and reliability.
4. Warehouse has product inventory.
5. Warehouse can distribute products to local store.
6. There will be monthly, weekly or daily sales report.
7. Customer can search in different stores for product availability.
8. Customer can see top selling products of specific category.

Platforms:

- Programming Language : PL/SQL
- IDE : Oracle 10g

Entity Relationship Diagram (ERD)

An entity relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities.

Elements of ERD:

- **Entity:** An entity is an object or concept about which you want to store information.
- **Relationship:** Relationship shows how entities share information in the database.
- **Attributes:** An attribute refers to a database component.

Entity Sets:

- Customer
- Branch
- Membership
- Product
- Transaction
- Warehouse
- Category

Entity and Table Set Name with Attributes and data types

Attributes of Branch:

- Branch_ID(primary key) - int
- Location - varchar(11)
- Phone - int

Attributes of Category:

- C_ID(primary key) – int
- C_Name – varchar(20)

Attributes of Customer:

- Cust_ID(primary key) - int
- M_ID(foreign key) - int
- C_Name - varchar(50)
- Email - varchar(50)

Attributes of Membership:

- M_ID(primary key) - int
- P_Range_From - number
- P_Range_To - number
- Discount_Rate – number
- Type - varchar(50)

Attributes of Product:

- P_ID(primary key) - int
- C_ID(foreign key)- int
- Selling_Price – number
- P_Name – varchar(50)

Attributes of Transaction:

- T_ID(primary key) - int
- Cust_ID(foreign key) – int
- Branch_ID(primary key) – int
- Total_Price – number
- Date – date

Attributes of Warehouse:

- S_ID(primary key) - int
- S_Date - date
- P_ID(foreign key) – int
- Buying_Price - number
- P_Quantity – int

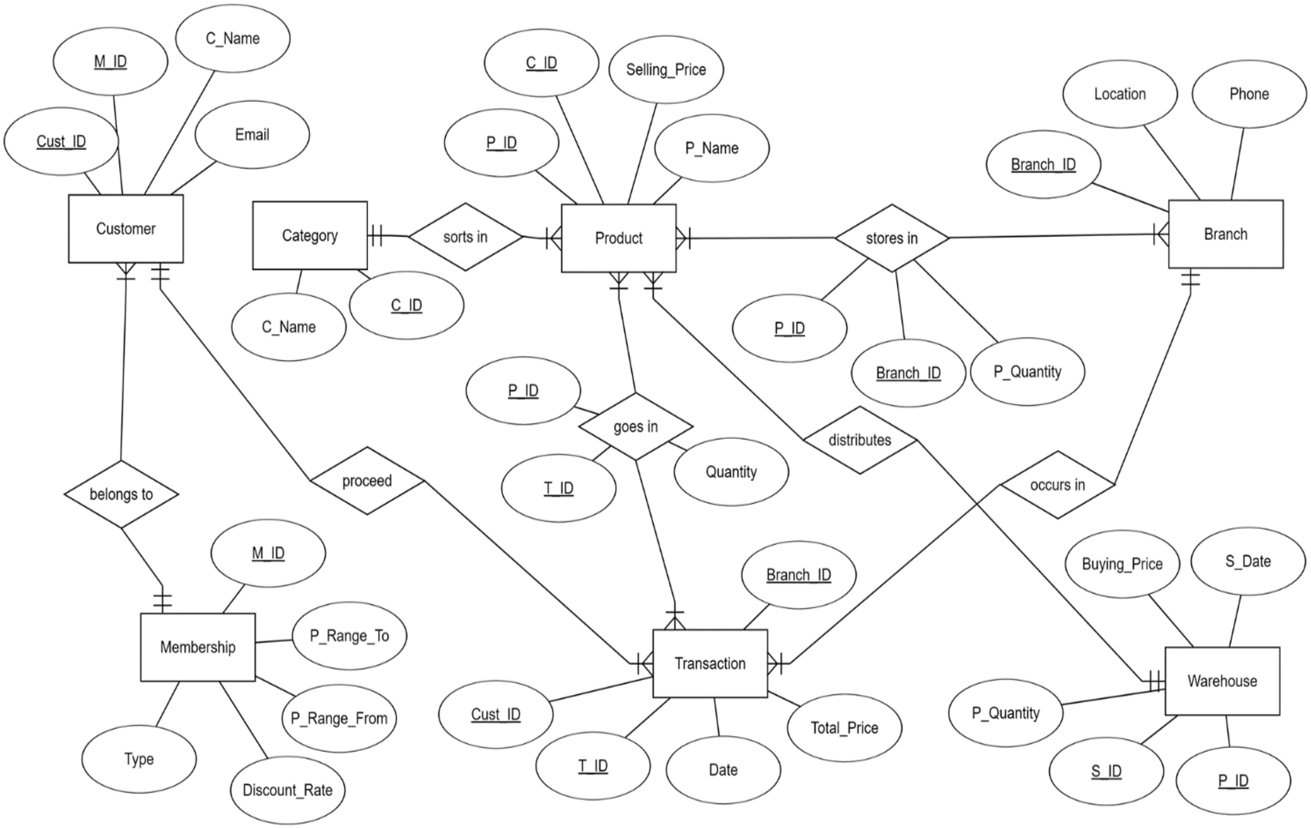
Attributes of goes in:

- P_ID(foreign key) - int
- T_ID(foreign key) – int
- Quantity – int

Attributes of stores in:

- P_ID(foreign key) - int
- Branch_ID(foreign key) – int
- P_Quantity – int

ERD for GO CART:



Relational Model with Fragmentation Schema:

Global Schema:

- Branch(branch_ID, Location, Phone)
- Category(C_ID, C_Name)
- Membership(M_ID, P_Range_From, P_Range_To, Discount_Rate, Type)
- Product(P_ID, Selling_Price, P_Name, C_ID)
- Customer(C_ID, C_Name, Email, M_ID)
- Transaction(T_ID, Cust_ID, Branch_ID, Total_Price, T_Date)
- goes_in(P_ID, T_ID, Quantity)
- occurs_in(Branch_ID, T_ID)
- proceed(T_ID,C_ID)
- stores_in(P_ID, P_Quantity, Branch_ID)
- warehouse(S_ID, S_Date, P_ID, P_Quantity, Buying_Price)

Fragmentation Schema with allocated site:

- Branch1 = $PJ_{branch_ID,Location,Phone}SL_{Location=’Dhanmondi’}$Branch
- Branch2 = $PJ_{branch_ID,Location,Phone}SL_{Location=’Mohammadpur’}$Branch
- Stores_in1 = $PJ_{P\_ID,\ P\_Quantity,\ Branch\_ID}SL_{Branch\_ID\ =\ 1}$ stores_in
- Stores_in2 = $PJ_{P\_ID,\ P\_Quantity,\ Branch\_ID}SL_{Branch\_ID\ =\ 2}$ stores_in
- Transaction1 = $PJ_{T\_ID,Cust\_ID,Branch\_ID,Total\_Price,T\_Date}SL_{Branch\_ID\ =\ 1}$ Transaction
- Transaction2 = $PJ_{T\_ID,Cust\_ID,Branch\_ID,Total\_Price,T\_Date}SL_{Branch\_ID\ =\ 2}$ Transaction
- Branch1 @ site1, Stores_in1 @ site1, Transaction1 @ site1
- Branch2 @ site2, Stores_in2 @ site2, Transaction2 @ site2

Database Profile:

- Here we can see, For Branch (Branch_ID, Location, Phone) relation the database profile contains the following information with fragments Branch$_1$, Branch$_2$:

Branch$_1$

card(Branch$_1$) = 1

site(Branch$_1$) = 1

	Branch_ID	Location	Phone
Size	16 bits	11 bits	16 bits
Val	1	1	1

Branch$_2$

card(Branch$_2$) = 1

site(Branch$_2$) = 2

	Branch_ID	Location	Phone
Size	16 bits	11 bits	16 bits
Val	1	1	1

- Here we can see, For Stores_in(P_ID, P_Quantity, Branch_ID) relation the database profile contains the following information with fragments Stores_in$_1$, Stores_in$_2$:

Stores_in$_1$

card(Stores_in$_1$): 1

site(Stores_in$_1$): 1

	P_ID	P_Quantity	Branch_ID
Size	16 bits	16 bits	16 bits
Val	1	1	1

Stores_in$_2$

card(Stores_in$_2$): 2

site(Stores_in$_2$): 2

	P_ID	P_Quantity	Branch_ID
Size	16 bits	16 bits	16 bits
Val	2	2	1

- Here we can see, For relation Transaction(T_ID, Cust_ID, Branch_ID, Total_Price, T_Date) the database profile contains the following information with fragments Transaction$_1$, Transaction$_2$:

Transaction$_1$

card(Transaction$_1$): 2

site(Transaction$_1$): 1

	T_ID	Cust_ID	Branch_ID	Total_Price	T_Date
size	16 bits	16 bits	16 bits	21 bytes	7 bytes
val	2	2	1	2	2

Transaction$_2$

card(Transaction$_2$): 2

site(Transaction$_2$): 2

	T_ID	Cust_ID	Branch_ID	Total_Price	T_Date
size	16 bits	16 bits	16 bits	21 bytes	7 bytes
val	2	2	1	2	2

Functions or Procedures(Given Five):

There are some functions and procedures we've added to our project.

1. **Top Selling Product:** We can see the top selling products of a week, a month or a year.
2. **Sales Report:** Sales report will be produced based on buy and sales.
3. **Warehouse Distribution:** Distribution data of the products on each branch will be kept in.
4. **Product availability :** Product availability of nearby branches will be informed to the customer.
5. **Membership:** Customer will get discount based on membership rank.

References:

1. Oracle Database 10G - The Complete Reference - Mcgraw Hill Osborne

Effects of Update: Here is given an example,

Effect of updating Type="Silver" of membership with M_ID = 1:

1st step:

membership1 (site1)

M_ID	P_Range_From	P_Range_To	Discount_Rate	Type
1	2000	10000	5	Bronze

membership2 (site2)

M_ID	P_Range_From	P_Range_To	Discount_Rate	Type
2	10000	20000	10	Silver
3	10000	20000	10	Silver

2nd Step:

Delete membership1 at site1 where M_ID=1;

membership1 (site1)

M_ID	P_Range_From	P_Range_To	Discount_Rate	Type
1	2000	10000	5	Bronze

membership2 (site2)

M_ID	P_Range_From	P_Range-To	Discount_Rate	Type
10002	10000	20000	10	Silver
10003	10000	20000	10	Silver

3rd Step:

Insert into membership2 (M_ID, P_ Range_From, P_Range_To, Discount_Rate, Type) at site 2 : (1, 10000, 20000, 10, “Silver”);

membership1 (site1)

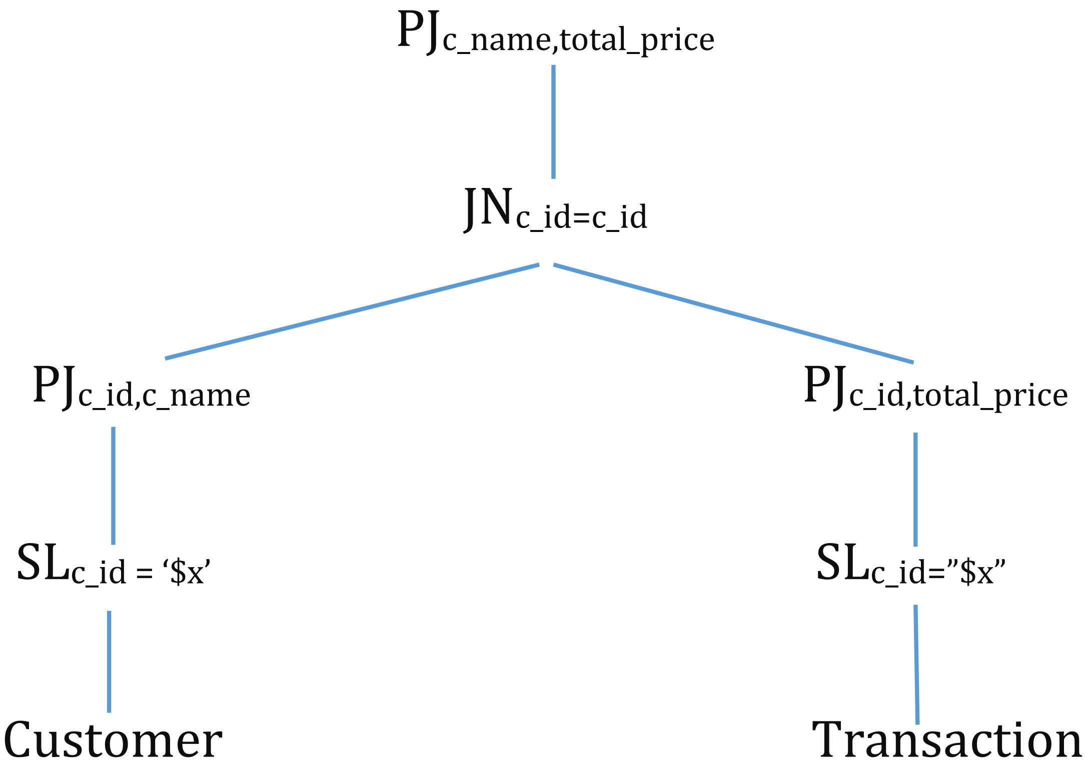
M_ID	P_Range_From	P_Range_To	Discount_Rate	Type

membership2 (site2)

M_ID	P_Range_From	P_Range_To	Discount_Rate	Type
2	10000	20000	10	Silver
3	10000	20000	10	Silver
1	10000	20000	10	Silver

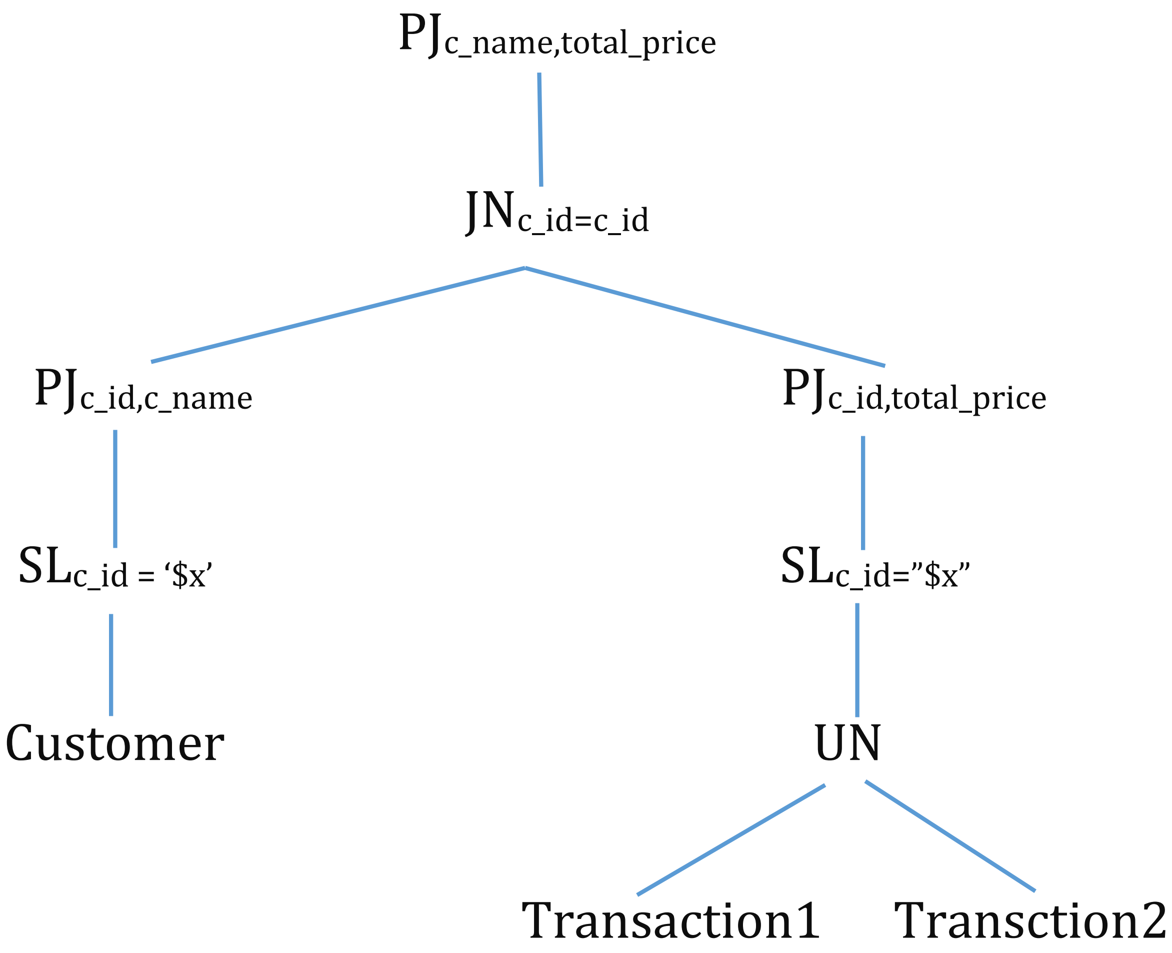
Operator Tree on Join: Here is given an example,

Query: $PJ_{c\_name,total\_price}((PJ_{c\_id,c\_name}SL_{c\_id='\$x'}\text{Customer})JN_{c\_id=c\_id}(PJ_{c\_id,total\_price}$ $SL_{c\_id='\$x'}\text{Transaction}))$



Operator Tree Using Canonical Expression:

Query: $PJ_{c\_name,total\_price}((PJ_{c\_id,c\_name}SL_{c\_id='\$x'}\text{Customer})JN_{c\_id=c\_id}(PJ_{c\_id,total\_price}$ $SL_{c\_id='\$x'}\text{(Transaction1 UN Transaction2)}))$



Qualified Relation: Here is given an example,

[transaction1: Total_Price<=1000]SJ$_{Branch\_ID=Branch\_ID}$ [branch1: Branch_ID=1]

Applying Rule 7,

- [transaction1SJ$_{Branch\_ID=Branch\_ID}$ branch1: Branch_ID= Branch_ID and Total_Price<=1000 and Branch_ID=1]

branch1

Branch_ID	Location	Phone
1	'Dhanmondi'	168552341

transaction1

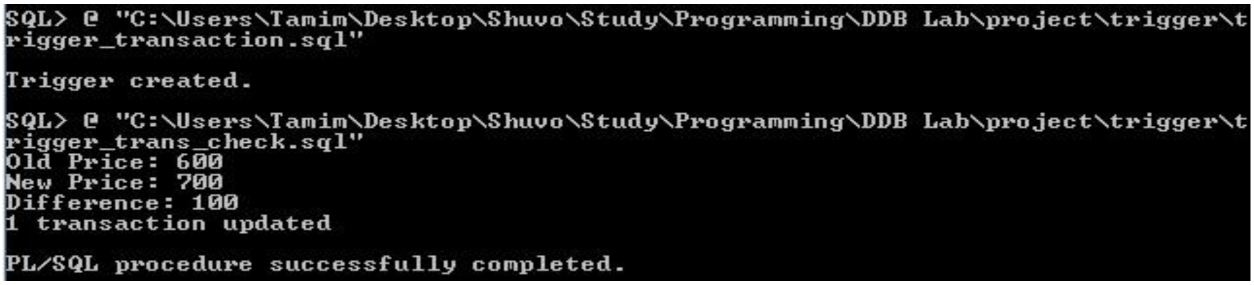
T_ID	Cust_ID	Branch_ID	Total_Price	T_Date
1	10001	1	500	‘9-JAN-18’
2	10002	1	2000	‘10-JAN-18’

After Semi-Joining

T_ID	Cust_ID	Quantity	Total_Price	T_Date
1	10001	1	500	‘9-JAN-18’

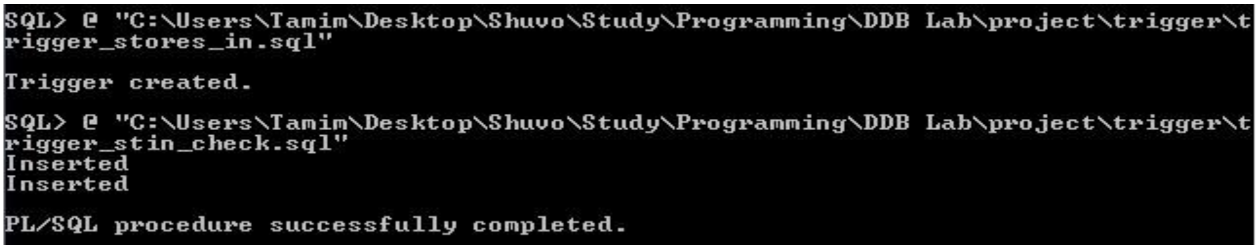
Trigger(Given three):

1. Trigger for updating total price in Transaction.

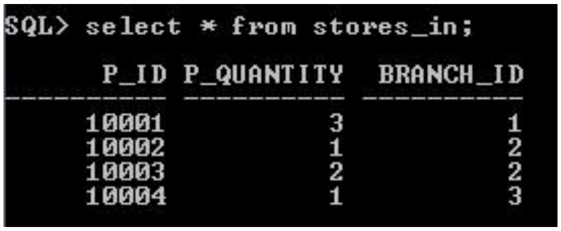


```
SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\t
rigger_transaction.sql"

Trigger created.

SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\t
rigger_trans_check.sql"
Old Price: 600
New Price: 700
Difference: 100
1 transaction updated

PL/SQL procedure successfully completed.
```

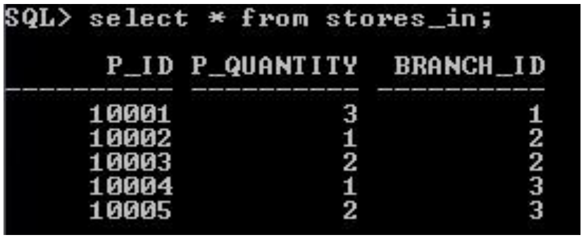
2. Trigger for inserting a value in Stores_in.



```
SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\t
rigger_stores_in.sql"

Trigger created.

SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\t
rigger_stin_check.sql"
Inserted
Inserted

PL/SQL procedure successfully completed.
```



```
SQL> select * from stores_in;

      P_ID P_QUANTITY  BRANCH_ID
---------- ---------- ----------
     10001          3          1
     10002          1          2
     10003          2          2
     10004          1          3
```

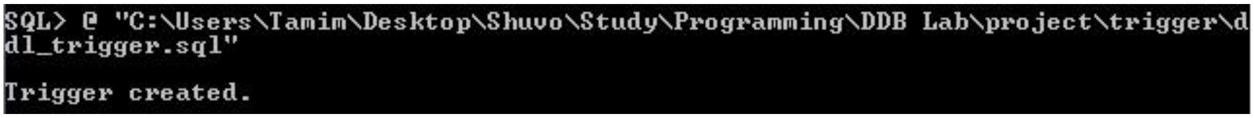


```
SQL> select * from stores_in;

      P_ID P_QUANTITY  BRANCH_ID
---------- ---------- ----------
     10001          3          1
     10002          1          2
     10003          2          2
     10004          1          3
     10005          2          3
```

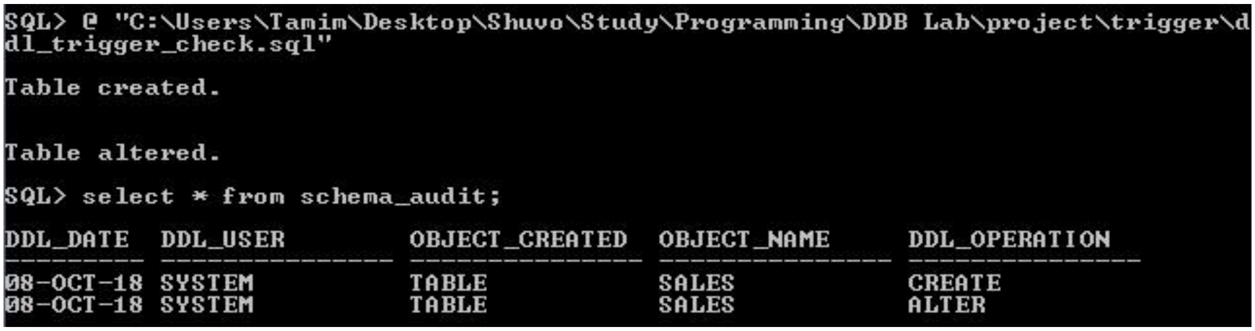
Before Insertion

After Insertion

3. DDL Trigger for Schema Audit.



```
SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\d
dl_trigger.sql"

Trigger created.
```



```
SQL> @ "C:\Users\Tamim\Desktop\Shuvo\Study\Programming\DDB Lab\project\trigger\d
dl_trigger_check.sql"

Table created.


Table altered.

SQL> select * from schema_audit;

DDL_DATE  DDL_USER        OBJECT_CREATED  OBJECT_NAME     DDL_OPERATION
--------- --------------- --------------- --------------- ---------------
08-OCT-18 SYSTEM          TABLE           SALES           CREATE
08-OCT-18 SYSTEM          TABLE           SALES           ALTER
```

Contribution:

There are several works we've done as a group and some of them as individual. Here some of my individual works are simply pointed:

- Create some queries for Functions and Procedures for several operations.
- Create some fragments for the sites.
- Distribute fragments to a site. As we are 3 group members, so we're using a host and 2 sites.
- Using VMWare for sites, perform several operations on fragments.
- Create DML and DDL triggers for insert, update, delete and audit operations.