Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No: CSE4126
Course Title: Distributed Database Lab
Project Name: Al-Bintum Super Shop

Submitted to:
Mohammd Imrul Jubair
Assistant Professor, AUST

Submitted by:
Oishee Bintey Hoque
Id: 15.01.04.005

Section: A (A1)

Group Members:
Al-Farabi Akash
Id: 15.01.04.010
Md. Saiful Islam
Id: 15.01.04.027

Project Summery:

Our project is on a **Super Shop Management System**. The project stores data in a distributed database management system for of all the necessary items required in a super shop.

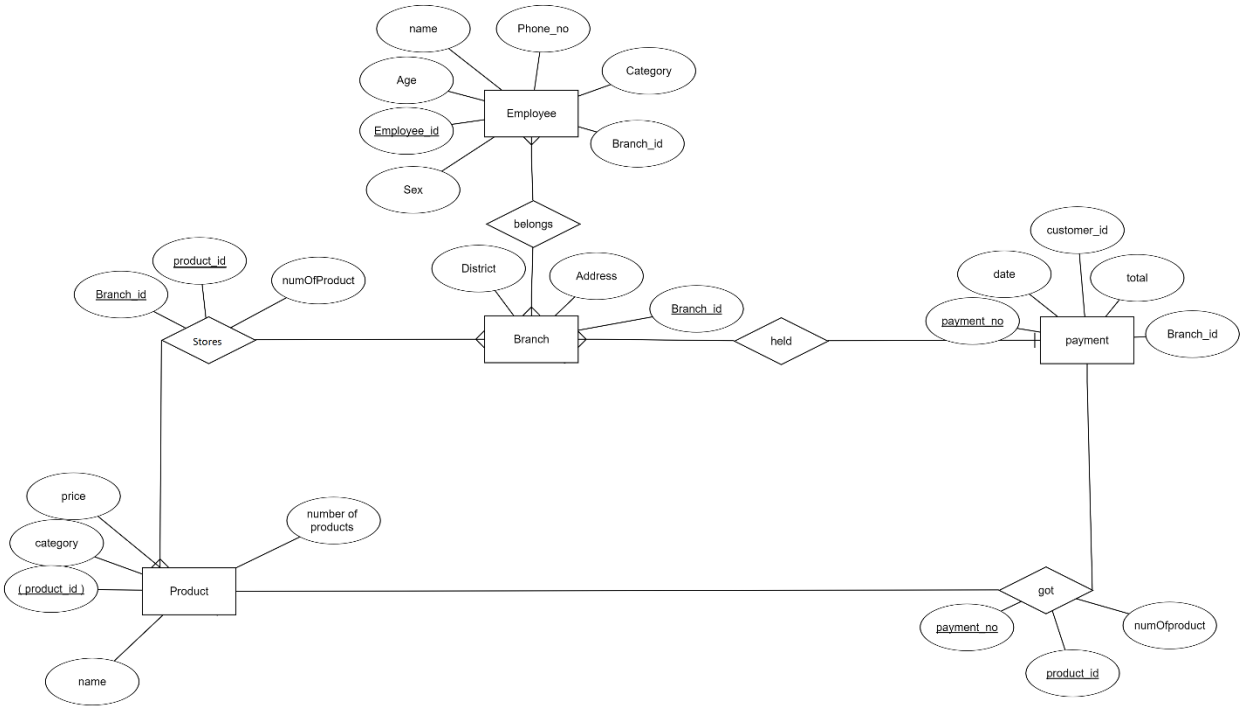
There are 6 tables in our database:

1. EMPLOYEE: Stores detailed information about the employees.
2. STORES: Stores information about which branch stores what products.
3. BRANCH: Stores branch details.
4. PRODUCT: Stores product details.
5. PAYMENT: Stores total bill of the customers.
6. GOT: Stores what products the customers bought.

Platforms:

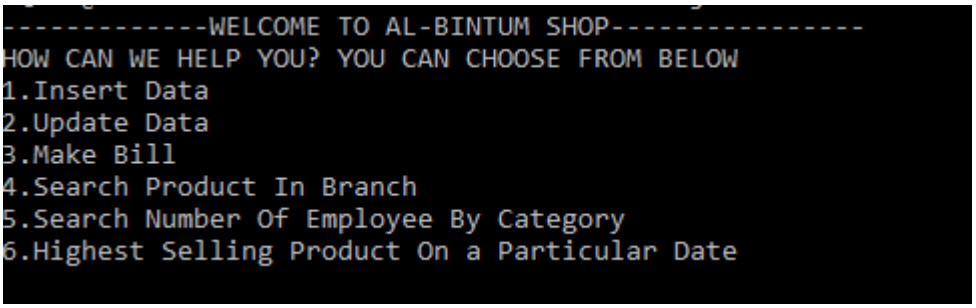
- Programming Language : PL/SQL
- IDE : Oracle 10g

Entity Relationship Diagram (ERD):



Overview of the system:

Our project is system for managing a super shop, which dynamically handles all the data. This super shop management system meticulously handles all the aspects related to the shop, i.e. sales, staff information, branch details and product availability. As the database is distributed to multiple sites, the risk of losing data due is loss in site is reduced and the system is improved.



My Contribution

1. Table and Fragmenation allocation of the Table.
2. Functionalities:
    a. Search Product by Product Id
    b. Update Employee Branch
    c. Selling Information Of a Branch
    d. Sequence
    e. Trigger

Global Relations:

Employee(employeeId ,ename,sex ,age,phoneNo,catagory,branchId**)**

Branch(branchId,address,district**)**

Product(productid,productname,price,category,numberofproducts**)**

Stores(productid,branchId,numberofproducts)

Got(paymentno,productid,noOfProducts**)**

Payment(paymentno,branchid,total,paymentdate**)**

Fragmentation Schema:

Employee1=$SL_{branchid=1}$(Employee)

Employee2=$SL_{branchid!=1}$(Employee)

Stores1= $SL_{branchid=1}Pj_{productid,branched,numberOfProducts}$(Stores)

Stores2= $SL_{branchid!=1}Pj_{productid,branched,numberOfProducts}$(Stores)

Branch1= $SL_{branchid!=1}$ (Branch)

Branch2= $SL_{branchid=1}$ (Branch)

Payment1= $SL_{branchid!=1}$ (Payment)

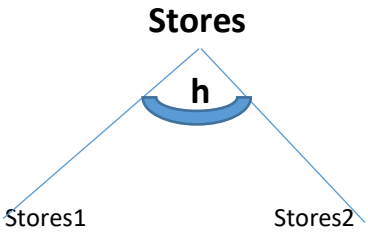
Payment2= $SL_{branchid=1}$ (Payment)

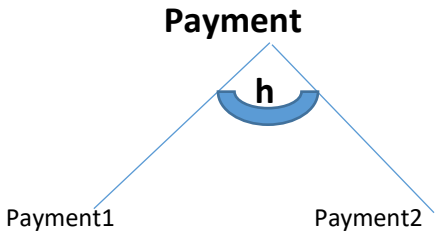
Allocation Schema:

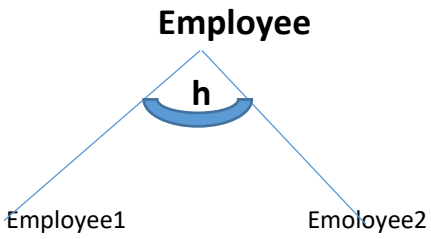
Site1(Host): Stores1,Product,Branch1,Employee1,Payment1,Got

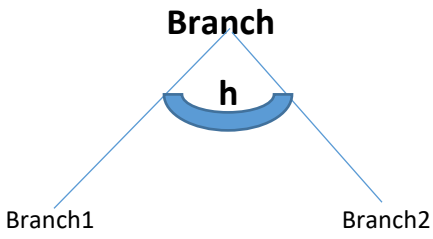
Site2: Stores2,Branch2,Employee2,Payment2

Fragmentation Tree:





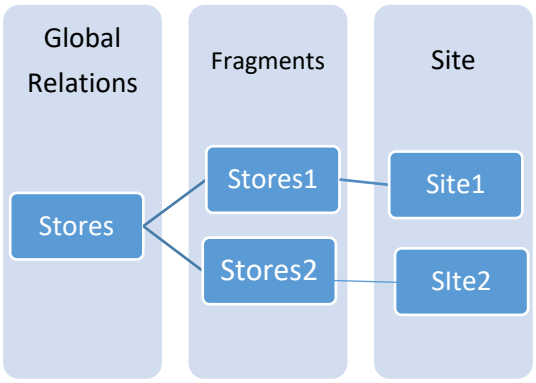




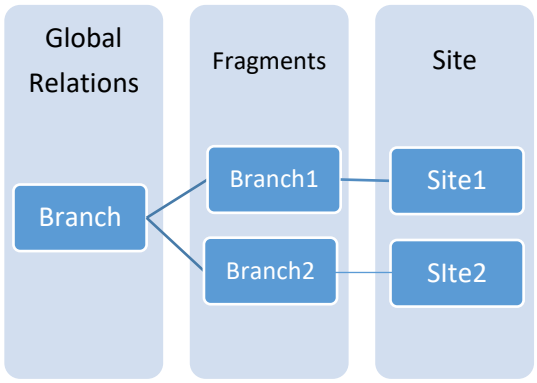
Allocation Schema (Visual Image):

Site1(Host): Stores1,Product,Branch1,Employee1,Payment1,Got

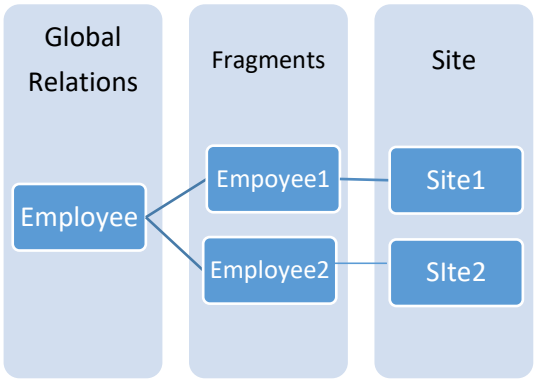
Site2: Stores2,Branch2,Employee2,Payment2



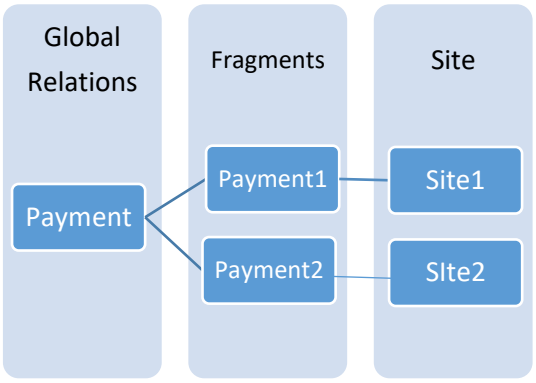
Stores Table



Branch Table



Employee Table



Payment Table

Database Profile For Store Table

Attributes of STORE

card (Store) = 19

site(Store) = Global

	branchId	ProductId	noOfProduct
Size	20	20	20
Val	3	9	19

card ($Store_1$) = 6

site($Store_1$) = 1

	branchId	ProductId	noOfProduct
Size	20	20	20
Val	1	6	6

card ($Store_2$) = 6

site($Store_2$) = 2

	branchId	ProductId	noOfProduct
Size	20	20	20
Val	2	6	13

Function:

Update Employee Branch:

Parameter: employeeId number, branchId number

Return: number

Description: takes employeeId,branchId and updates employee table according to branchId.

Exception: NONE

Example Update Process:

Update->Branch Id=2 where EmployeeId=10

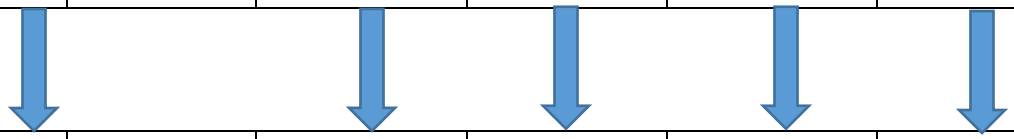
- Store corresponding data of employeeid.

-Insert in Employee2 With Updated Value

-Delete data from employee1

Employee1:

Employee Id	Branch Id	Ename	Sex	Phone	Address
10	1	Oishee	F	01xxxxx	Dhaka



Employee2:

Employee Id	Branch Id	Ename	Sex	Phone	Address
10	2	Oishee	F	01xxxxx	Dhaka

Employee1:

Employee Id	Branch Id	Ename	Sex	Phone	Address
~~10~~	~~1~~	~~Oishee~~	~~F~~	~~01xxxxx~~	~~Dhaka~~

Procedure:

Search Product Information By Id:

Parameter: productId number

Return: number

Description: takes productId and returns product information and availability in branch

Exception: NONE

Exaample:

```
Enter Product id:1
ProductId ProductName Price BranchId Quantity
3          t-shirt     1093        1     0
3          t-shirt     1093        2     68
3          t-shirt     1093        3     0
```

Sql Command:

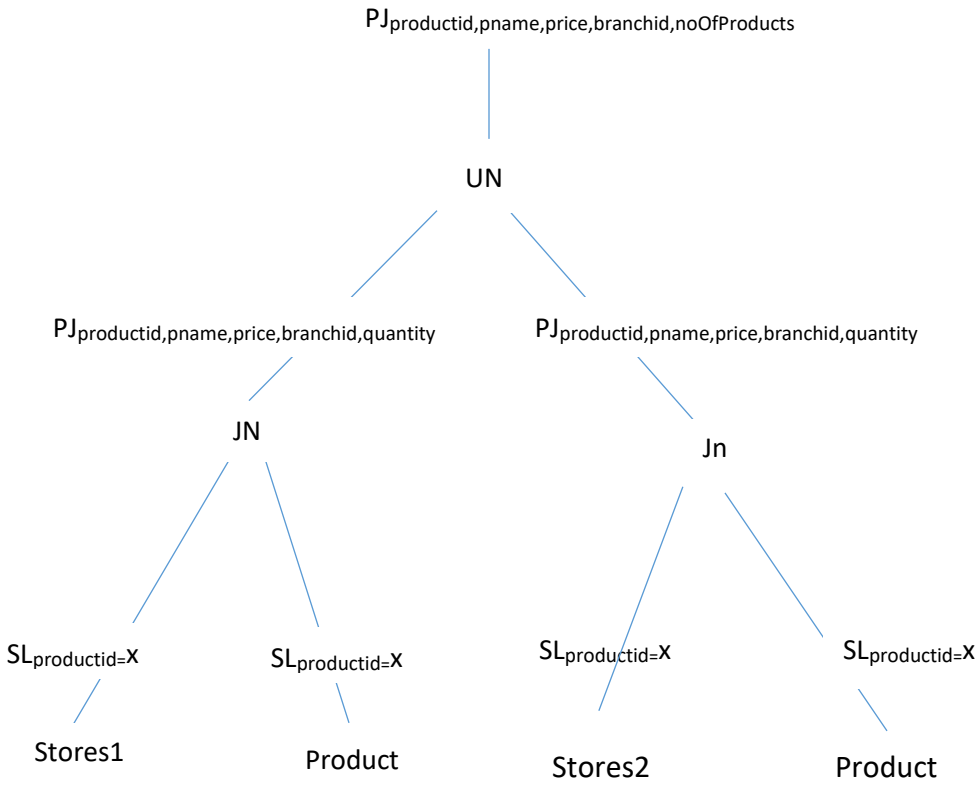
SELECT * from

((Select product.productid,pname,price,branchid,numberOfProducts from product Inner Join Store@site1 On product.productid=store1.productid where product.productid=3)

UNION

(Select product.productid,pname,price,branchid,numberOfProducts from product Inner Join Store@Site2 On product.productid=store2.productid where product.productid=3));

Operator Tree:



Function:

CREATE BILL:

Parameter: payment number

Return: number

Description:

-takes payment number

-Gernates the total bill and also individual total of each product

-insert data into payment table

-while creating bill, product quantity also gets updated in stores table.

-show message if any product is unavailable

Exception: NONE

Procedure:

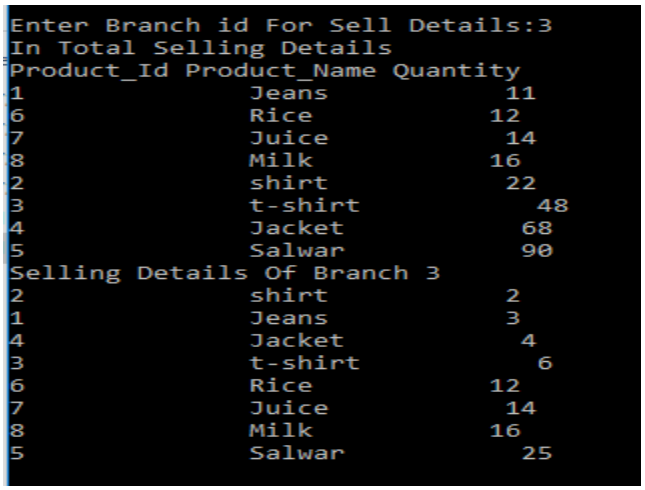
Search Sold Product Information By Branch Id:

Parameter: BranchId number

Return: number

Description: takes branchId, provides all the information of products which have been sold on that particular branch and also provides information on all sold products of all branch in total.

Exception: NONE

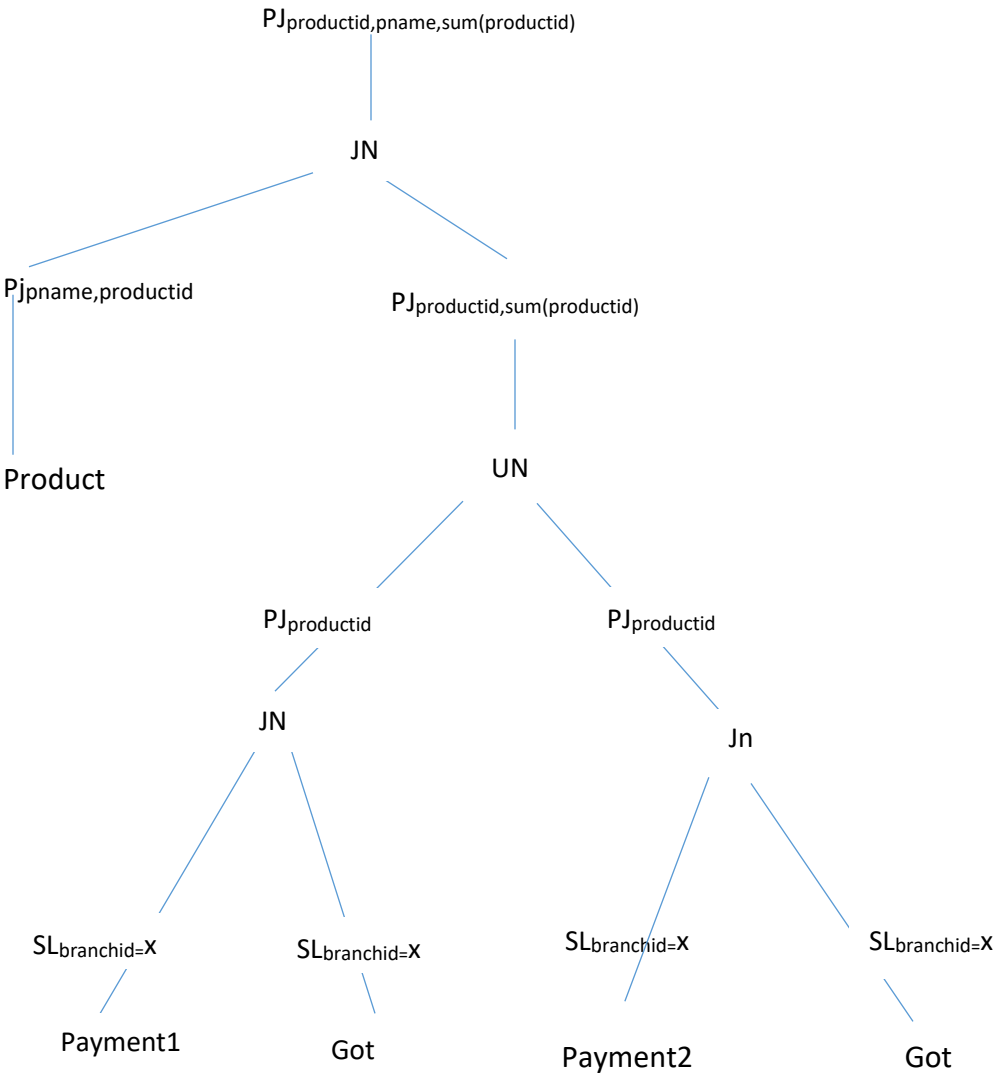


```
Enter Branch id For Sell Details:3
In Total Selling Details
Product_Id Product_Name Quantity
1               Jeans           11
6               Rice           12
7               Juice           14
8               Milk           16
2               shirt           22
3               t-shirt           48
4               Jacket           68
5               Salwar           90
Selling Details Of Branch 3
2               shirt           2
1               Jeans           3
4               Jacket           4
3               t-shirt           6
6               Rice           12
7               Juice           14
8               Milk           16
5               Salwar           25
```

Sql Command:

select pp.productid,pname as Product_Name,pp.p as Quantit from productinner join(select productid,sum(productid) as p from got group by productid) PP ON PP.productid=product.productid order by Quantity;

Operator Tree:



Number of Employee By Category:

Parameter: category varch

Return: number

Description:

- Returns Number of employee of given category

Exception: NONE

Package:

add_PriceAlter_Package:

Description: we put a procedure in the package. In that procedure which is inside the package updates price on given range and given percentage.

Example:

```
Enter Percentage of Price Increasing:2
Enter Min Price for Increasing:2000
Old Price of Jeans : 1929 BDT
New Price of Jeans : 1968 BDT
Price difference: 39
```

Sequence:

- Sequence has been used for keeping track of payment number.

Triggers:

Four After Input Trigger for Tables: Stores, Payment, Employee, branch

- Update values in necessary sites when global database gets any new input.
- Example: When user **gives input** in **Employee/Stores/Payment/Branch**, Input trigger of **Employee/Stores/Payment/Branch receives the data** and also **insert the data in its fragmented tables in desired sites**.
- Example:
  Data has been inserted into store table and with the trigger data has also been inserted into stores table at site2 as data with BranchId!=1 belongs to the store table of site2.

```
SQL> insert into STORES (branchId, productid, noOfProduct) values(3, 11, 10);
New Value Inserted In Site 2 Store Table

1 row created.
```

Global ‘Store’ Table

```
SQL> select * from stores;

  BRANCHID  PRODUCTID NOOFPRODUCT
---------- ---------- -----------
         3          6           4
         3          7           2
         3          8           8
         3          1           2
         3          3           0
         3          5           2
         1          1           2
         1          2           7
         1          3           0
         1          6          10
         1          5           6

  BRANCHID  PRODUCTID NOOFPRODUCT
---------- ---------- -----------
         1          4           9
         2          3          68
         2          4           0
         2          5           0
         2          1           0
         2          2           0
         2          8          10
         3         11          10
```

Store Table at site2.

```
SQL> select * from store2 @site2;

  BRANCHID  PRODUCTID NUMBEROFPRODUCTS
---------- ---------- ----------------
         3          6                4
         3          7                2
         3          8                8
         3          1                2
         3          3                0
         3          5                2
         2          3               68
         2          4                0
         2          5                0
         2          1                0
         2          2                0

  BRANCHID  PRODUCTID NUMBEROFPRODUCTS
---------- ---------- ----------------
         2          8               10
         3         11               10
```

Product Price After Update Trigger for Table: Product

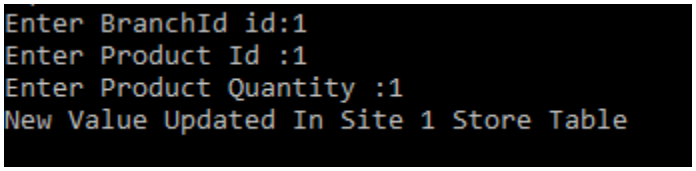
- Shows the old price, updated price and difference between the price
- Example:

```
Enter Percentage of Price Increasing:2
Enter Min Price for Increasing:2000
Old Price of Jeans : 1929 BDT
New Price of Jeans : 1968 BDT
Price difference: 39
Old Price of shirt : 1715 BDT
New Price of shirt : 1749 BDT
Price difference: 34
Old Price of t-shirt : 1072 BDT
New Price of t-shirt : 1093 BDT
Price difference: 21
Old Price of Salwar : 1286 BDT
New Price of Salwar : 1312 BDT
Price difference: 26
Old Price of Juice : 1887 BDT
New Price of Juice : 1925 BDT
Price difference: 38
Old Price of Milk : 735 BDT
New Price of Milk : 750 BDT
Price difference: 15

PL/SQL procedure successfully completed.
```

Store Product After Update Trigger for Table: Product

- If any store updates its number of products, this trigger updates the value in its fragmented tables and also shows a message which site got the update.
- Example:



Conclusion & Future Plan

As we improved a total system for super shop and nowadays it is spreading rapidly so we have the opportunity to develop it further.

1. Adding more option in database for simplifying (site wise).
2. Creating more usable functions or procedures that can will be more efficient in data handling.
3. Using the machine learning process for taking long-term decision.