CSE

4126

DISTRIBUTED DATABASE MANAGEMENT SYSTEM LAB

FINAL PROJECT REPORT

Restaurant Management System

SUBMITTED BY

KAZI HASIB AHMED | 15-01-04-135

Summary:

Restaurants are increasing rapidly in Bangladesh. Many popular restaurant are opening new branches. New branches means new customers and new data. Therefore, our project focuses on reliable and easy management of all the branches by merging new branches data with existing branches data. We have designed this distributed database management system for Restaurant Company.

We have done things which are given below:

- Horizontal and vertical fragmentation.
- Allocation of the fragmentations.
- Distribution Transparency.
    - Level-1
    - Level-2
    - Level-3
- Steps of effect of update.
- Query Transformation.
- Basic functions of Restaurant management (like- Bill, Cart, Total sales of a day etc.)
- Triggers.

Entity set:

- branch - Keeps information of the branches of the restaurant.
- customer - Keeps information of the customers.
- employee- Keeps information of the employees.
- type- Keeps employee types and their information
- foodCategory – Keeps information of the food categories.
- foodItem- Keeps information of the food items.
- makeOrder- Keeps information of the customer orders.
- orderDetails- Keeps all the ordered items details of a customer order.
- ingredients- Keeps information of the ingredients.
- Ing_details- Keeps details ingredients purchased and remained.
- Ing_daily_used- Keeps details of the ingredients, which have been used.

Global Schema:

- branch(branch_id, branch_name, branch_address, phone)
- customer(customer_id, branch_id, name , phone)
- employee(employee_id, type_id, branch_id, employee_name, age, address, sex, email, phone)
- type(type_id, type_name, salary)
- foodCategory(c_id, c_name)
- foodItem(food_id, category_id, f_name, price, discount, f_details)
- makeOrder(order_id, customer_id, amount, order_date)
- orderDetail(order_id, food_id, quantity)
- ingredients(i_id, name)
- ing_details(entry_id, i_id, branch_id, total_cost, quantity_remained, entry_date, expiry_date, quantity_brought)
- ing_daily_use(use_id, entry_id, used_date, quantity_used)

Fragmentation Schema:

- $employee_1 = employee\ SJ_{employee.branch\_id=branch.branch\_id}\ SL_{branch\_name='Dhaka'}\ branch$
- $employee_2 = employee\ SJ_{employee.branch\_id=branch.branch\_id}\ SL_{branch\_name='Comilla'}\ branch$
- $employee_3 = employee\ SJ_{employee.branch\_id=branch.branch\_id}\ SL_{branch\_name='Chittagong'}\ branch$

- $ing\_details_1 = PJ_{entry\_id,\ i\_id,\ branch\_id,\ total\_cost,\ quantity\_remained}\ ing\_details$
- $ing\_details_2 = PJ_{entry\_id,\ entry\_date,\ expiry\_date,\ quantity\_brought}\ ing\_details$

Allocation Schema:

There are two sites in our project. (Named as host and site_link)

- $employee_1$ @ site_link
- $employee_2$ @ host, site_link
- $employee_3$ @ host

- $ing\_details_1$ @ host
- $ing\_details_2$ @ site_link

Function/Procedure/Trigger:

We have done functions, procedures and triggers in 3 different parts.

- Employee.
- Customer and Order.
- Ingredient.

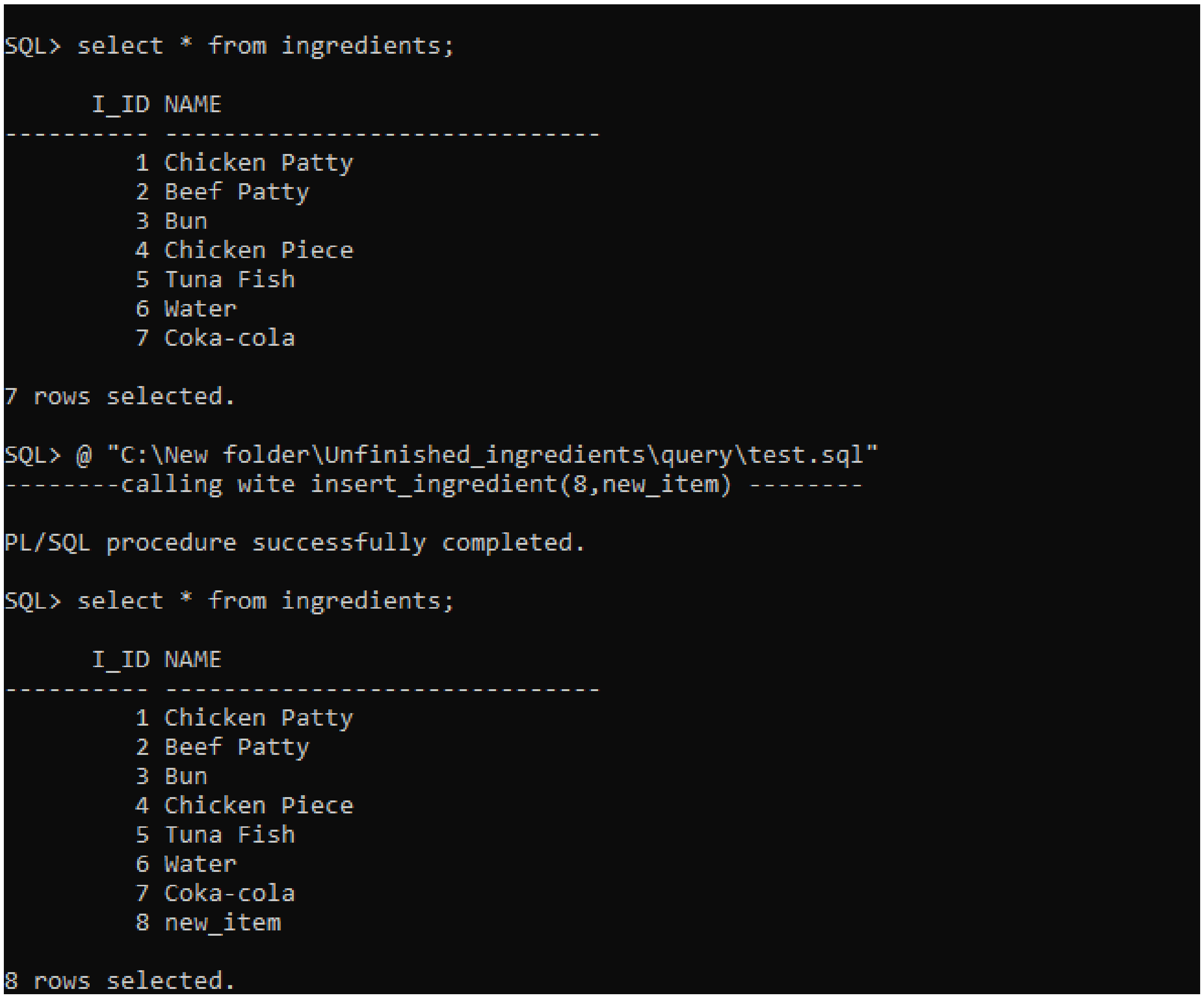
I have done the ingredient part from these. Therefore, I have given the details of all the functions, procedures and triggers related to that part. You will find others function, trigger and procedure details in other members reports.

Ingredient:

- **insert_ingredient:**

  This inserts an ingredient data.

  - Input: Ingredient id and name.
  - Output: Confirmation of Insert.
  - Transparency Level: 1.

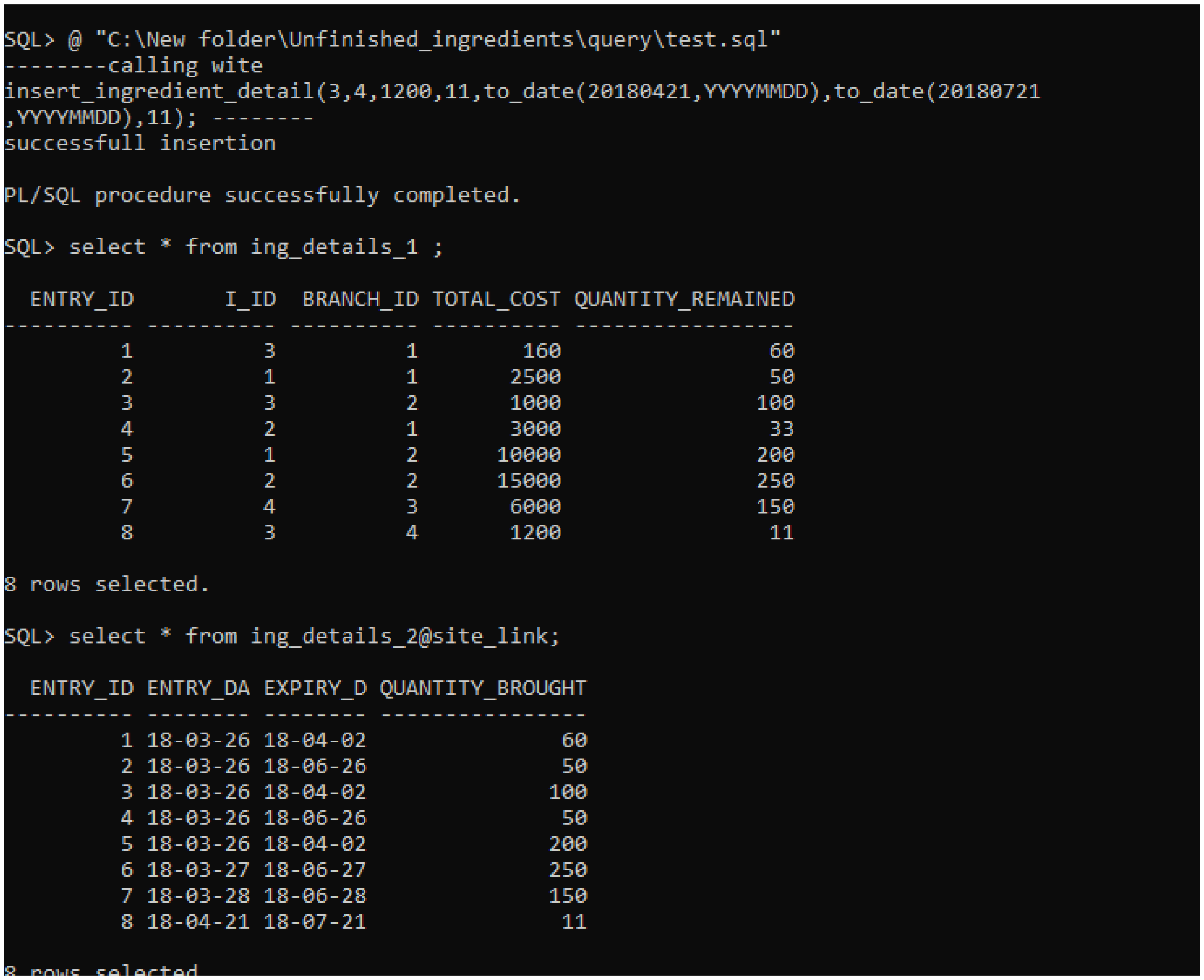


```
SQL> select * from ingredients;

      I_ID NAME
---------- ------------------------------
         1 Chicken Patty
         2 Beef Patty
         3 Bun
         4 Chicken Piece
         5 Tuna Fish
         6 Water
         7 Coka-cola

7 rows selected.

SQL> @ "C:\New folder\Unfinished_ingredients\query\test.sql"
--------calling wite insert_ingredient(8,new_item) --------

PL/SQL procedure successfully completed.

SQL> select * from ingredients;

      I_ID NAME
---------- ------------------------------
         1 Chicken Patty
         2 Beef Patty
         3 Bun
         4 Chicken Piece
         5 Tuna Fish
         6 Water
         7 Coka-cola
         8 new_item

8 rows selected.
```

- **insert_ingredient_detail:**

  This inserts an ingredient purchased details to its two vertical fragments.

  - Input: Ingredient id, branch id, total cost,, entry date, expiry date, quantity brought and quantity remained.
  - Output: Confirmation of Insert.
  - Transparency Level: 1.

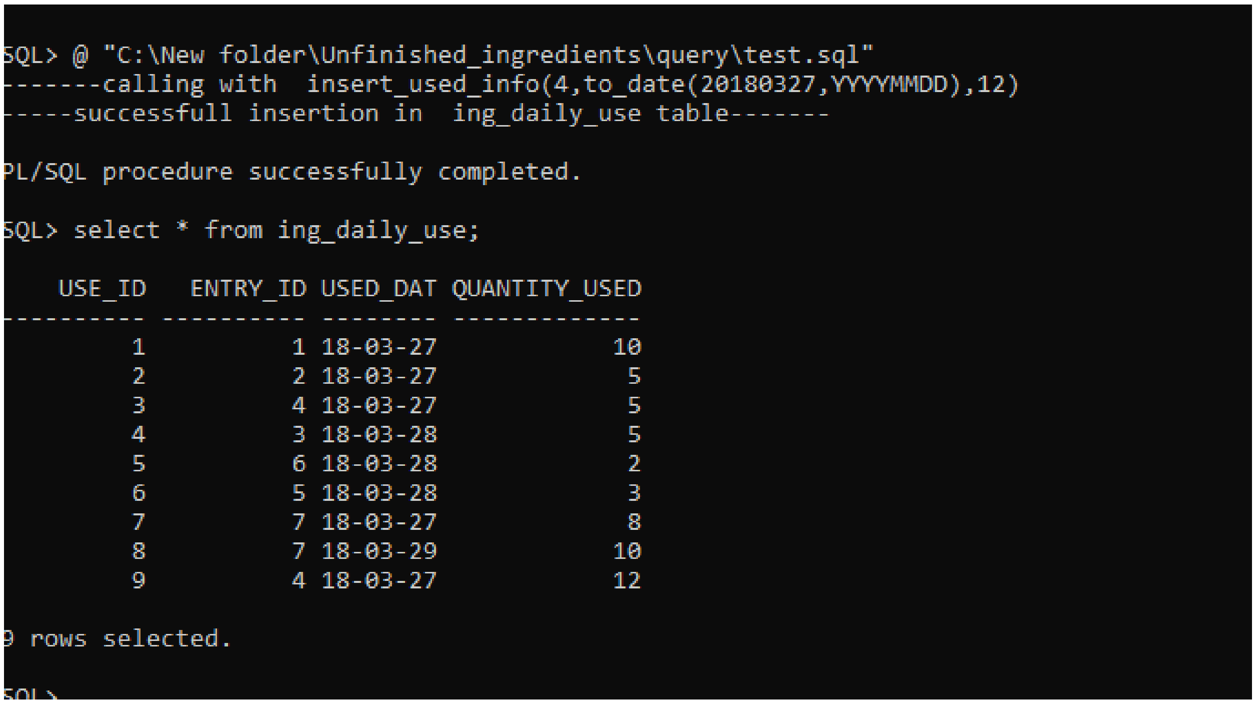


```
SQL> @ "C:\New folder\Unfinished_ingredients\query\test.sql"
--------calling wite
insert_ingredient_detail(3,4,1200,11,to_date(20180421,YYYYMMDD),to_date(20180721
,YYYYMMDD),11); --------
successfull insertion

PL/SQL procedure successfully completed.

SQL> select * from ing_details_1 ;

  ENTRY_ID       I_ID  BRANCH_ID TOTAL_COST QUANTITY_REMAINED
---------- ---------- ---------- ---------- -----------------
         1          3          1        160                60
         2          1          1       2500                50
         3          3          2       1000               100
         4          2          1       3000                33
         5          1          2      10000               200
         6          2          2      15000               250
         7          4          3       6000               150
         8          3          4       1200                11

8 rows selected.

SQL> select * from ing_details_2@site_link;

  ENTRY_ID ENTRY_DA EXPIRY_D QUANTITY_BROUGHT
---------- -------- -------- ----------------
         1 18-03-26 18-04-02               60
         2 18-03-26 18-06-26               50
         3 18-03-26 18-04-02              100
         4 18-03-26 18-06-26               50
         5 18-03-26 18-04-02              200
         6 18-03-27 18-06-27              250
         7 18-03-28 18-06-28              150
         8 18-04-21 18-07-21               11

8 rows selected
```

- **insert_used_info:**

  This inserts an ingredient used details and calculating remaining quantity and insert it on ‘ing_details$_1$’ fragment table on its sites.

  - Input: Entry id, used date and quantity used.
  - Output: Confirmation of Insert.
  - Transparency Level: 1.

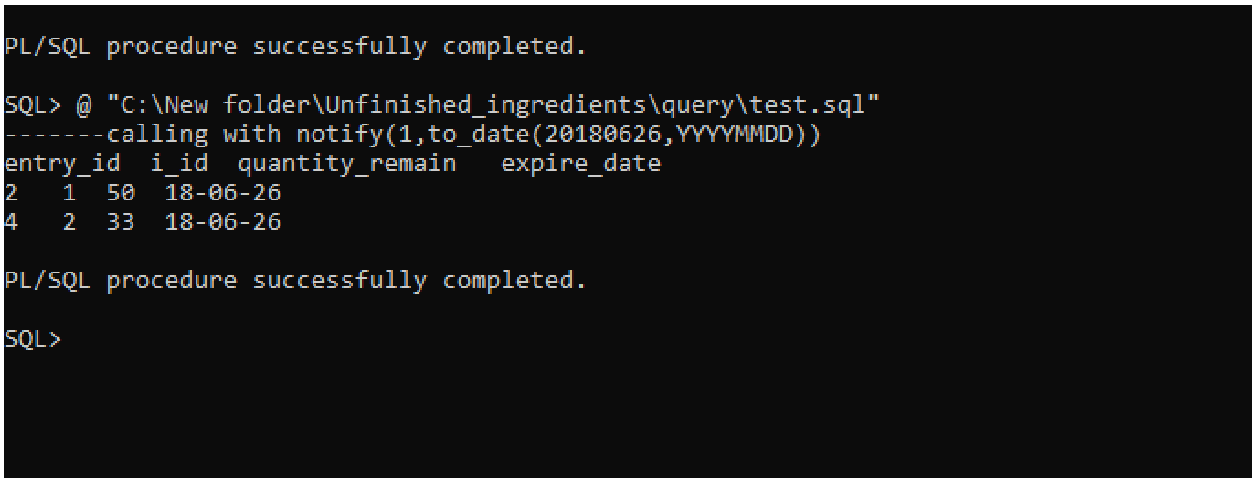


```
SQL> @ "C:\New folder\Unfinished_ingredients\query\test.sql"
-------calling with  insert_used_info(4,to_date(20180327,YYYYMMDD),12)
-----successfull insertion in  ing_daily_use table-------

PL/SQL procedure successfully completed.

SQL> select * from ing_daily_use;

    USE_ID   ENTRY_ID USED_DAT QUANTITY_USED
---------- ---------- -------- -------------
         1          1 18-03-27            10
         2          2 18-03-27             5
         3          4 18-03-27             5
         4          3 18-03-28             5
         5          6 18-03-28             2
         6          5 18-03-28             3
         7          7 18-03-27             8
         8          7 18-03-29            10
         9          4 18-03-27            12

9 rows selected.

SQL>
```

- **notify_expiry_date:**

  This function shows user which ingredients will be expired in a specific date for a specific branch.

  - Input: Branch id and date.
  - Output: Entry id, ingredient id, quantity remained and expiry date.
  - Transparency Level: 3.

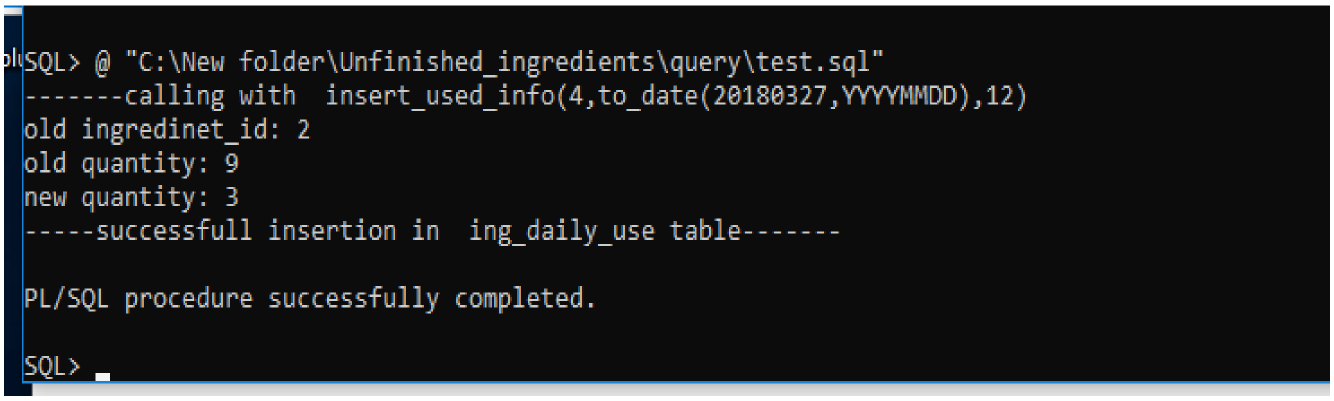


```
PL/SQL procedure successfully completed.

SQL> @ "C:\New folder\Unfinished_ingredients\query\test.sql"
-------calling with notify(1,to_date(20180626,YYYYMMDD))
entry_id  i_id  quantity_remain   expire_date
2   1  50  18-06-26
4   2  33  18-06-26

PL/SQL procedure successfully completed.

SQL>
```

➢ **trigger_before_update_ing_details_1:**

When we insert data in ing_daily_use, it updates the remaining quantity of ing_details_1.This trigger function will show old remaining quantity and new remaining quantity before updating ing_details_1.

- Input: No input.
- Output: Shows old and new value of the remaining_quantity.
- Transparency Level: 2.



```
SQL> @ "C:\New folder\Unfinished_ingredients\query\test.sql"
-------calling with  insert_used_info(4,to_date(20180327,YYYYMMDD),12)
old ingredinet_id: 2
old quantity: 9
new quantity: 3
-----successfull insertion in  ing_daily_use table-------

PL/SQL procedure successfully completed.

SQL> _
```

<u>Employee</u>:

- delete_employee_type_checks_fk.
- delete_employee1_site1.
- insert_employee.
- transfer_employee.
- trigger_employee2_update_type_id.
- update_employee2_type_id.
- employees_name_given_type.
- employees_given_type_and_branch.

<u>Customer and Order</u>:

- amount_sale_in_a_date.
- find_customers.
- find_customers_of_a_date.
- get_order_info.
- insert_order.
- make_bill.

<u>Query Transformation</u>:

Here is a query which we have transformed into its fragments. It is used in the 'notify_expiry_date' function.

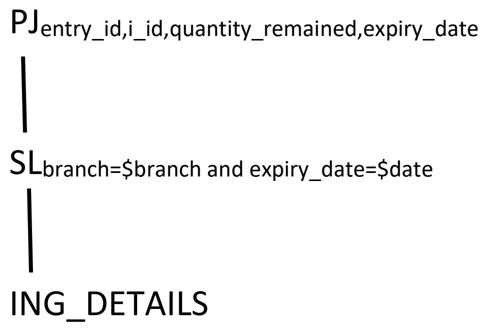
Initial query:

```
select entry_id,i_id,quantity_remained,expiry_date from ing_details where
branch_id=$branch and expiry_date=to_date($date,'YYYYMMDD');
```

Query (Q_{Global}):

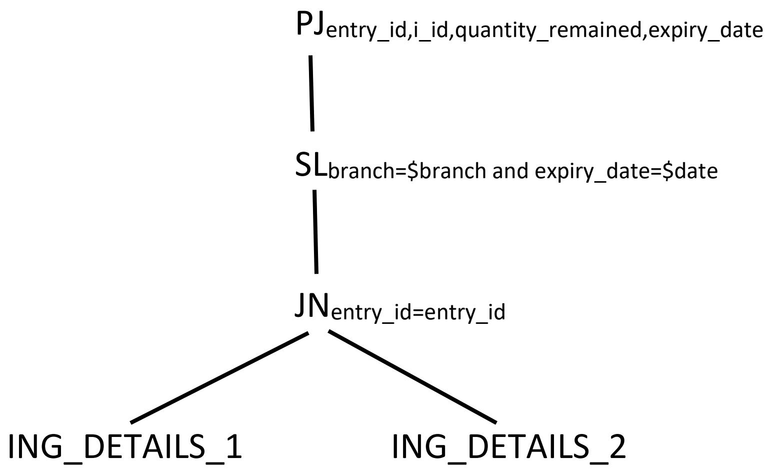
$PJ_{\text{entry\_id,i\_id,quantity\_remained,expiry\_date}}\ SL_{\text{branch=\$branch and expiry\_date=\$date}}\ \text{ING\_DETAILS}$

Operator Tree (T_{global}):



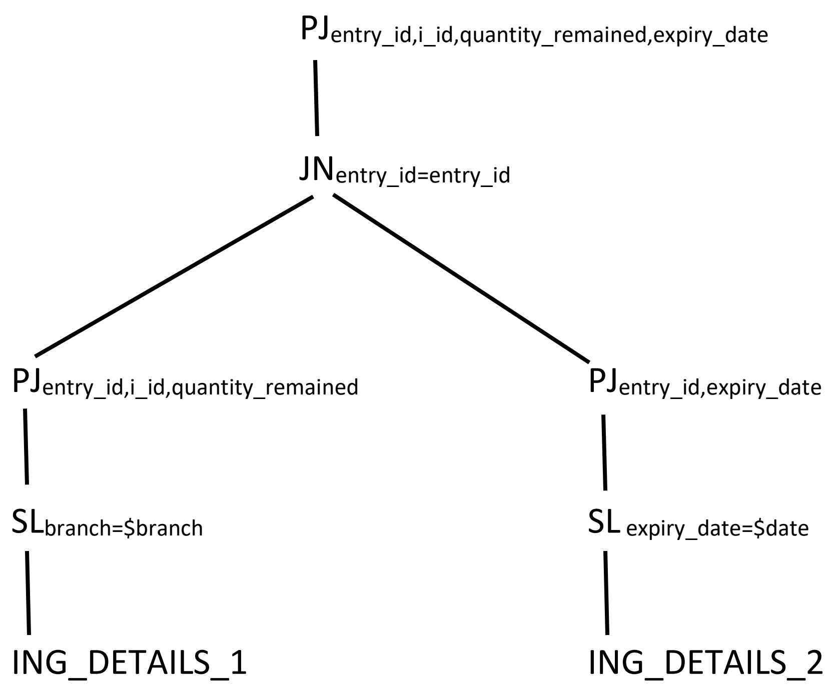
$PJ_{entry\_id,i\_id,quantity\_remained,expiry\_date}$
$SL_{branch=\$branch\ and\ expiry\_date=\$date}$
ING_DETAILS

($T_{fragments}$):



$PJ_{entry\_id,i\_id,quantity\_remained,expiry\_date}$
$SL_{branch=\$branch\ and\ expiry\_date=\$date}$
$JN_{entry\_id=entry\_id}$
ING_DETAILS_1
ING_DETAILS_2

After Appling Criteria 1 and 2 ($T_{transformed}$):



Transformed Query ($Q_{transformed}$):

$PJ_{entry\_id,i\_id,quantity\_remained,expiry\_date}$ ($PJ_{entry\_id,i\_id,quantity\_remained}$ $SL_{branch=\$branch}$ ING_DETAILS_1 $JN_{entry\_id=entry\_id}$ $PJ_{entry\_id,expiry\_date}$ $SL_{expiry\_date=\$date}$ ING_DETAILS_2)

Final Query:

select t1.entry_id,i_id,quantity_remained,expiry_date from (select entry_id,i_id,quantity_remained from ing_details_1 where branch_id=$branch) t1

inner join (select entry_id,expiry_date from ing_details_2@site_link where expiry_date=$date) t2 on t1.entry_id=t2.entry_id;

Effect of update:

I have shown a scenario which will show effect of update for fragmentation which has been done on this project.

Problem: Transfer an employee to 'Dhaka' branch who's employee id= 4.

Solution:

Employee has three fragments ($employee_1$ for Dhaka, $employee_2$ for Comilla, $employee_3$ for Chittagong).

Now, employee id= 4 has branch_id=3 ('Comilla') is fragmented as $employee_2$ which is allocated on both the sites and 'Dhaka' (branch_id=1) is fragmented as $employee_1$ which is allocated on host.

Step-1: Store the necessary data from any site of $employee_2$.

$employee_2$ @ host

employee_id	type_id	employee_name	Age	address	Sex	email	phone
4	1	Habib	33	Mirpur-2	M	Habib101@gmail.com	01884838292

Step-2: Insert the stored data to $employee_1$.

$employee_1$ @ site_link

employee_id	type_id	branch_id	employee_name	Age	address	Sex	email	phone
4	1	1	Habib	33	Mirpur-2	M	Habib101@gmail.com	01884838292

Step-3: Delete the data of that employee from all sites employee$_2$.

employee$_2$ @ host

employee_id	type_id	branch_id	employee_name	Age	address	Sex	email	phone
4	1	3	Habib	33	Mirpur-2	M	Habib101@gmail.com	01884838292

employee$_2$ @ site_link

employee_id	type_id	branch_id	employee_name	Age	address	Sex	email	phone
4	1	3	Habib	33	Mirpur-2	M	Habib101@gmail.com	01884838292

Note: Green is used to show inserted data and Red is used to show deleted data.

Conclusion:

This project helped us to understand how to work on DDB system. By this project we have learned how to connect PCs and use their database to from one to another. We have also learned how to design DDB for its user need. We get the view of how to make a DDB efficient. This will encourage us to further work on DDB system.