بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Data warehousing and Big Data

ASSIGNMENT TWO

Semester 2, 2018

Submitted by

Muhammad Kundi

Student Id: 15924072

Table of Contents

Project Overview

New World is one of the largest supermarket chains in NZ. The stores are placed all over the country. New World has thousands of customers and thus it's important for the organisation to analyse the behaviour of their customers. As a result of such analysis ,NewWorld can optimise their selling strategies e.g. by having relevant promotions on different product.

This task can be achieved by a Data Warehouse where customer's transactions from different Data Sources(DSs) are stored in the DW on daily basis. The Data Integration in this process can be maintained by ETL (Extraction,Transformation and Loading). But the data generated from DSs is not in the format required by DW. Therefore, data needs to be processed in the transformation layer of ETL using the information from Master Data as shown in Figure-1

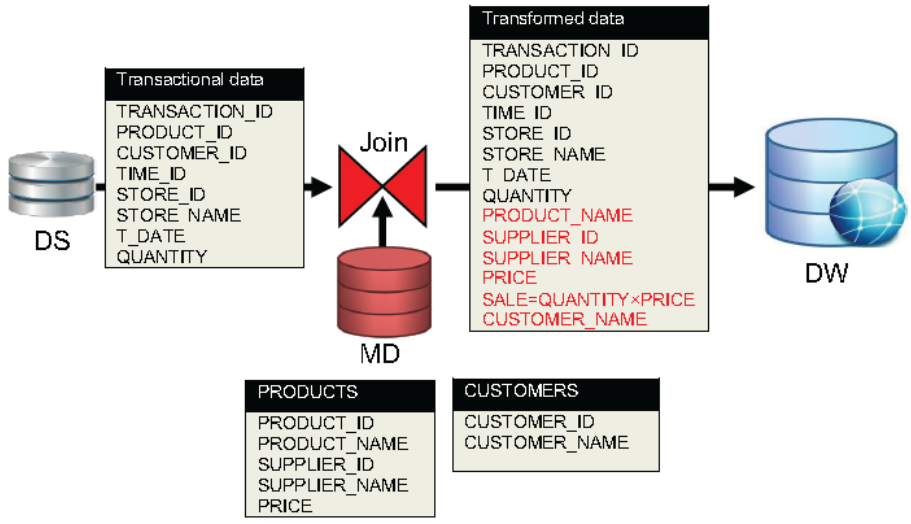

Figure 1 An example of enriching transactional data with information from MD

Data and structure specification

There are 10,000 records in DS. This data will be generated randomly based on 100 products, 50 customers, 10 stores, and one year time period as a date - from 01-Jan-17 to 31-Dec-17. The values for the quantity attribute will be random between 1 and 10. The other two tables named PRODUCTS and CUSTOMERS in MD with 100 and 50 records respectively

Pseudocode for INLJ

Following steps are taken for this task

1. Making a procedure called INLJ
2. Make new variables to facilitate the curser run in batch form

Serial No	Variable Names
1.	variable_total_records
2.	variable_records_per_batch
3.	variable_total_batch
4.	variable_count_batch
5.	variable_from
6.	variable_to

3. Declaring those variables which are holding values for customer and product from Master Data at the moment when the transaction records are joined.

Serial No	Variable Names
1.	variable_customer_name
2.	variable_product_name
3.	variable_supplier_id
4.	variable_supplier_name
5.	variable_price

4. Making a new variable called : var_count
5. Get all transactions for transaction table and assigning it to var_total_records;
6. Using a while to get records in batches of 100
7. Print the information to user also
8. Declaring a new cursor :cur_transaction.
9. The For loop is actually reading the data tuple by tuple using the cursor
10. Assign customer name in var_customer_name
11. At the same time doing pre existing check of data.
12. Find the product information from Master Data and then set these product name, supplier id, supplier name, and price attributes value in variable_product_name, variable_supplier_id , variable_supplier_name , variable_price variables respectively.
13. Inserting new record if the customer is not existing in the d_customers dimention table
14. Adding new record for product if product is not existing in the d_products dimension table. Similar operation is doen for d_stores,d_suppliers,d_time dimension tables.
15. Fact record is inserted in w_facts fact table in DW If the same fact record id is not already existing in dimension table. Inserting record on w_facts table if the same record is not already existing
16. Assigning the variables value to NULL
17. Variable_count value is reset to 1.
18. The For loop is ended
19. Commit
20. Again we are re assigning the variables to take the next 100 records .
21. And finally ending procedure INLJ

OLAP queries

```
--------------------------------------------------------------------------------
--Question No-1. Which product generated maximum sales in September, 2017?
--------------------------------------------------------------------------------

SELECT d_products.product_name,
       SUM ( w_facts.sale ) sale,
       DENSE_RANK ( ) OVER (ORDER BY SUM ( w_facts.sale ) DESC NULLS LAST)
          RANK
  FROM w_facts,
       d_products,
       (SELECT d_time.time_id
          FROM d_time
         WHERE cal_month = 'September' AND cal_year = 2017) v_dt
 WHERE w_facts.product_id = d_products.product_id
   AND w_facts.time_id = v_dt.time_id
GROUP BY d_products.product_name;
```

Output-1

SQL | Fetched 50 rows in 0.07 seconds

	PRODUCT_NAME	SALE	RANK
1	Corn	2710.62	1
2	Pears	2201.31	2
3	Lemon / Lime juice	1930.6	3
4	Celery	1676.34	4
5	Tofu	1435.48	5
6	Ginger	1426.8	6
7	Pasta	1363.35	7
8	Tomatoes	1299.75	8
9	Oregano	1248	9
10	Melon	1214.95	10
11	Applesauce	1175.05	11
12	Mac and cheese	1173	12
13	Bouillon cubes	1124.68	13
14	Ice cream / Sorbet	1119.8	14
15	Black pepper	1040	15
16	Mint	988.68	16
17	Grapefruit	955.34	17
18	Asparagus	912	18
19	Soy sauce	885.5	19
20	Soups	882	20

```
--------------------------------------------------------------------------------
--Question No-2. Determine top three supplier names based on highest sales of their products.
--------------------------------------------------------------------------------

SELECT *
  FROM (SELECT DENSE_RANK ( )
                  OVER (ORDER BY SUM ( w_facts.sale ) DESC NULLS LAST)
                  RANK,
               d_suppliers.supplier_name,
               SUM ( w_facts.sale ) sale
          FROM w_facts, d_suppliers
         WHERE w_facts.supplier_id = d_suppliers.supplier_id
        GROUP BY d_suppliers.supplier_name)
 WHERE RANK < 4;
```

Output-2

	RANK	SUPPLIER_NAME	SALE
1	1	A.G. Edwards Inc.	100845.38
2	2	The AES Corporation	58503.3
3	3	CellStar Corp.	45817.28

```
 --------------------------------------------------------------------------------
--Question No-3. Determine the top 3 store names who generated highest sales in September, 2017.
--------------------------------------------------------------------------------

SELECT *
  FROM (SELECT DENSE_RANK ( )
                  OVER (ORDER BY SUM ( w_facts.sale ) DESC NULLS LAST)
                  RANK,
               d_stores.store_name,
               SUM ( w_facts.sale ) sale
          FROM w_facts,
```

```
                d_stores,
                (SELECT d_time.time_id
                   FROM d_time
                  WHERE cal_month = 'September' AND cal_year = 2017) v_dt
          WHERE w_facts.time_id = v_dt.time_id
            AND w_facts.store_id = d_stores.store_id
         GROUP BY store_name)
 WHERE RANK < 4;
```

Output-3

	RANK	STORE_NAME	SALE
1	1	West Auckland	7933.36
2	2	St. james	6967.84
3	3	Massey	6958.78

```
--------------------------------------------------------------------------------
--Question No-4. Presents the quarterly sales analysis for all stores using drill down query
concepts.
--------------------------------------------------------------------------------

SELECT d_stores.store_name,
       SUM ( DECODE ( d_time.cal_quarter, 1, w_facts.sale, 0 ) ) q1_2017,
       SUM ( DECODE ( d_time.cal_quarter, 2, w_facts.sale, 0 ) ) q2_2017,
       SUM ( DECODE ( d_time.cal_quarter, 3, w_facts.sale, 0 ) ) q3_2017,
       SUM ( DECODE ( d_time.cal_quarter, 4, w_facts.sale, 0 ) ) q4_2017
  FROM w_facts, d_stores, d_time
 WHERE d_stores.store_id = w_facts.store_id
   AND d_time.time_id = w_facts.time_id
GROUP BY d_stores.store_name
ORDER BY d_stores.store_name;
```

Output-4

	STORE_NAME	Q1_2017	Q2_2017	Q3_2017	Q4_2017
1	Albany	20331.79	18920.54	18753.29	20068.59
2	East Auckland	20352.88	18731.22	20151.15	18624.58
3	Henderson	9927.44	11170.7	11170.68	10127.25
4	Manukau	20770.48	18719.46	19759.99	18691.07
5	Massey	19739.52	22494.63	20648.2	19757.01
6	Queen St.	9984.35	10538.95	9962.53	9953.42
7	St. james	18319.02	19285.43	18852.92	21023.03
8	West Auckland	19964.71	19178.95	19309.41	21222.97
9	Westgate	23644.5	19618.32	18339.86	19349.73
10	Whangaparaora	18980.28	20407.19	19191.85	19678.87

```
--------------------------------------------------------------------------------
------ Question No-5. Create a materialised view with name “STORE_PRODUCT_ANALYSIS” that presents
store
----------     and product wise sales. The results should be ordered by store name and then product
name
```

```
CREATE MATERIALIZED VIEW STORE_PRODUCT_ANALYSIS
AS
SELECT s.store_name "Store Name", dp.product_name "Product Name", SUM(wf.sale) "Sale"
  FROM w_facts wf
 INNER JOIN d_products dp ON wf.product_id = dp.product_id
 INNER JOIN d_stores ds ON wf.store_id = s.store_id
 GROUP BY ROLLUP(ds.store_name, dp.product_name)
```

```
 ORDER BY ds.store_name, dp.product_name;

SELECT * FROM store_product_analysis;
```

Output-5

	Store Name	Product Name	Sale
1	Albany	Apples	456.32
2	Albany	Applesauce	1191.6
3	Albany	Asparagus	712.5
4	Albany	Avocados	425.28
5	Albany	BBQ sauce	645.12
6	Albany	Bagels	427.35
7	Albany	Baked beans	580.77
8	Albany	Bananas	732.45
9	Albany	Basil	385.48
10	Albany	Berries	199.64
11	Albany	Black pepper	800
12	Albany	Bouillon cubes	1505.62
13	Albany	Breakfasts	514.47
14	Albany	Broccoli	1045.74
15	Albany	Burritos	900.77
16	Albany	Carrots	169.88
17	Albany	Cauliflower	655.88
18	Albany	Celery	1751.4
19	Albany	Cereal	810.9
20	Albany	Cherries	1146.37

```
--------------------------------------------------------------------------------------------
------Question No-6. Create a materialised view with name “MONTH_STORE_ANALYSIS” that presents month
-----and store wise sales. The results should be ordered by month name and then store name.
--------------------------------------------------------------------------------------------

CREATE MATERIALIZED VIEW MONTH_STORE_ANALYSIS
AS
SELECT dt.cal_month "Month", ds.store_name "Store Name", SUM(wf.sale) "Sale"
  FROM w_facts f
 INNER JOIN d_time dt ON wf.time_id = dt.time_id
 INNER JOIN d_stores ds ON wf.store_id = ds.store_id
 GROUP BY ROLLUP(dt.cal_month, ds.store_name)
 ORDER BY dt.cal_month, ds.store_name;

 SELECT * FROM month_store_analysis;
```

Output-6

	Month	Store Name	Sale
1	April	Albany	6988.89
2	April	East Auckland	6946.96
3	April	Henderson	3870.42
4	April	Manukau	5966.31
5	April	Massey	7567.34
6	April	Queen St.	3268.84
7	April	St. james	8064.01
8	April	West Auckland	5951.41
9	April	Westgate	6274.81
10	April	Whangaparaora	6707.94
11	April	(null)	61606.93
12	August	Albany	6784.9
13	August	East Auckland	7455.58
14	August	Henderson	3072.27
15	August	Manukau	5562.8
16	August	Massey	5771.53
17	August	Queen St.	3297.62
18	August	St. james	6409.33
19	August	West Auckland	6379.86
20	August	Westgate	6502.87

Summary

In the above task, we can clearly find out the differences between normal databses and DW.We can perform daily routine operations on the normal data bases which are well known as 'Relational Data Bases'. We also call them as OLTP i.e Online Transaction Processing database because of the continuous transactions involved.But in RDBMS we do not apply analytical methods as it can be very costly for the database operation.

For the analytics, we need DW where we can easily execute OLAP (online analytical processing). Data warehouse is structured to access historical data and make analytics fast .

In Normal databases we use relational models but In this project we have used star schema for data modelling.Star Schema is normally used for DW. This technique is used to map multidimensional decision support into a relational database. The star schema represents aggregated data for specific business activities. Using this schema, one can store aggregated data from multiple sources that will represent different aspects of business operations.

We also noticed that in order to convert the transactional data to the format required by DW we need to enrich data according to DW specifications. The join operator used for this enrichment of data will be Index Nested Loop Join (INLJ).This is a join operator between DS and MD.In INLJ ,DS is scanned in batches of tuples and based on each of the tuple in a batch, the disk-based MD is accessed using an index on the join attribute. This is shown in Figure-2 where tuple from the batch of DS's is combined with the tuple from MD.

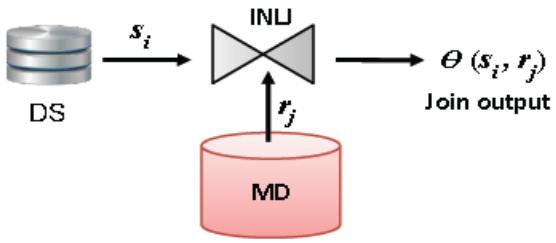

Figure 2 Execution architecture of INLJ

Finally using INLJ after we have transferred all the data in DW using batches of 100 members each, we applied the OLAP queries to analyse the data.