A Report for Building and Analysing the Data Warehouse for Countdown Stores in NZ

Prepared for:	Muhammad Asif Naeem
Prepared by:	Mao Chuan Li
Student ID:	14854389
Submit Date:	2015/09/12
Paper Name:	Data Warehousing and Big Data
Paper Number:	COMP810

1 Project Overview

Countdown is one of the biggest supermarkets spreading across New Zealand with 175 stores and 18,000 staff to serve more than 2.5 million customers every week. So they have accumulated tons of transaction data and user information. With these valuable information, Countdown could create a data warehouse to centrally integrate all business information for business analysts to query and analyse the latest data and generate reports based on them.

This project is to design and create such a data warehouse system with a randomly generated 10, 000 transaction records and 100 products. Due to the data from transaction data is incomplete for the desired data warehouse data structure, a complementary master data with detailed product and supplier information is used. When joining the transaction data and master product data, the most famous Meshjoin algorithm is harnessed as shown in following section 3. The classical star schema is used to model the data structure in the data warehouse, which is shown in section 4. After all 10,000 transaction records were imported into the data warehouse, 5 requested OLAP queries were executed on the data warehouse and the output is listed in section 5. At last, section 6 listed some lessons learned from this project.

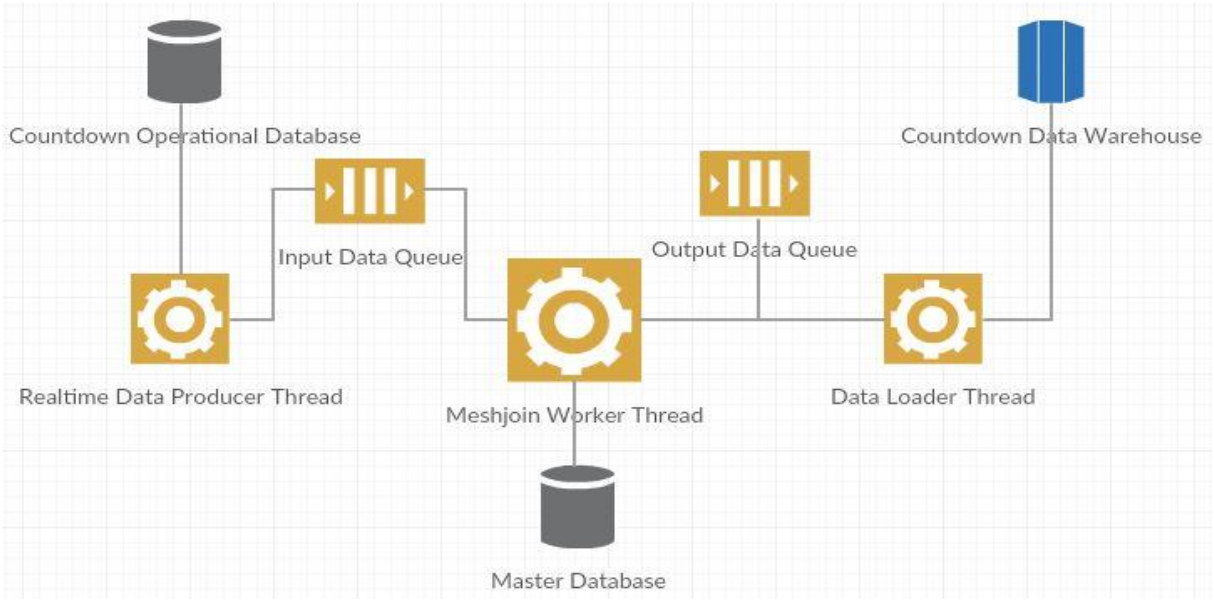
2 Countdown DW Project Architecture

The following diagram shows the general architecture of the project. Two logical relational databases are used for storing the transaction data and the data warehouse data. The database connections are specified in the system.properties configuration file with the following keys (2 connections may link to the same DB):

```
# The operationl database connection;
in.db.driverClassName=org.apache.derby.jdbc.ClientDriver
in.db.url=jdbc:derby://localhost:1527/countdownDB
in.db.username=kqc3001
in.db.password=password
# The data warehouse database connection;
out.db.driverClassName=org.apache.derby.jdbc.ClientDriver
out.db.url=jdbc:derby://localhost:1527/countdownDW
out.db.username=kqc3001
out.db.password=password
```

To work with the 2 databases, the following 3 threads are running concurrently and independently from each other:

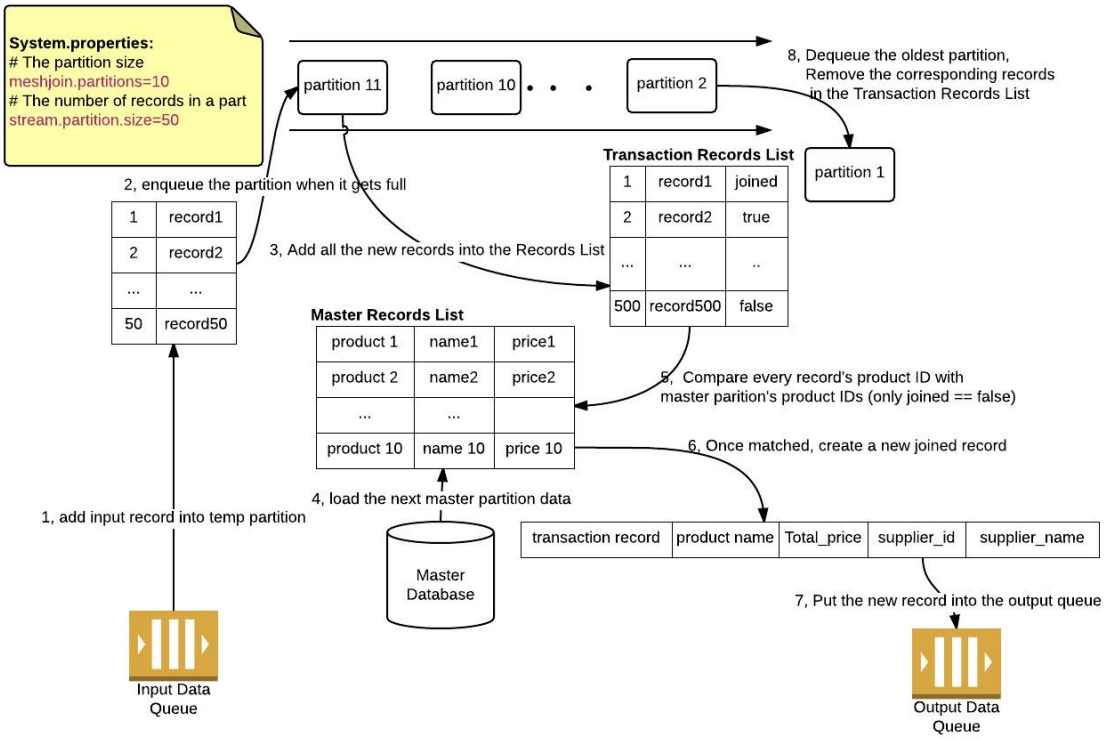
1. **Realtime Data Producer Thread** – which extracts the transaction data one by one and put it into the global “Input Data Queue”.
2. **Meshjoin Worker Thread** – which reads in the transaction records from the global input queue, and create a new joined record and put the new record in the global “Output Data Queue”, using the famous Meshjoin algorithm.
3. **Data Loader Thread** – which reads in the new joined records from the “Output Data Queue”, and writes the data into Data Warehouse according to the star schema.



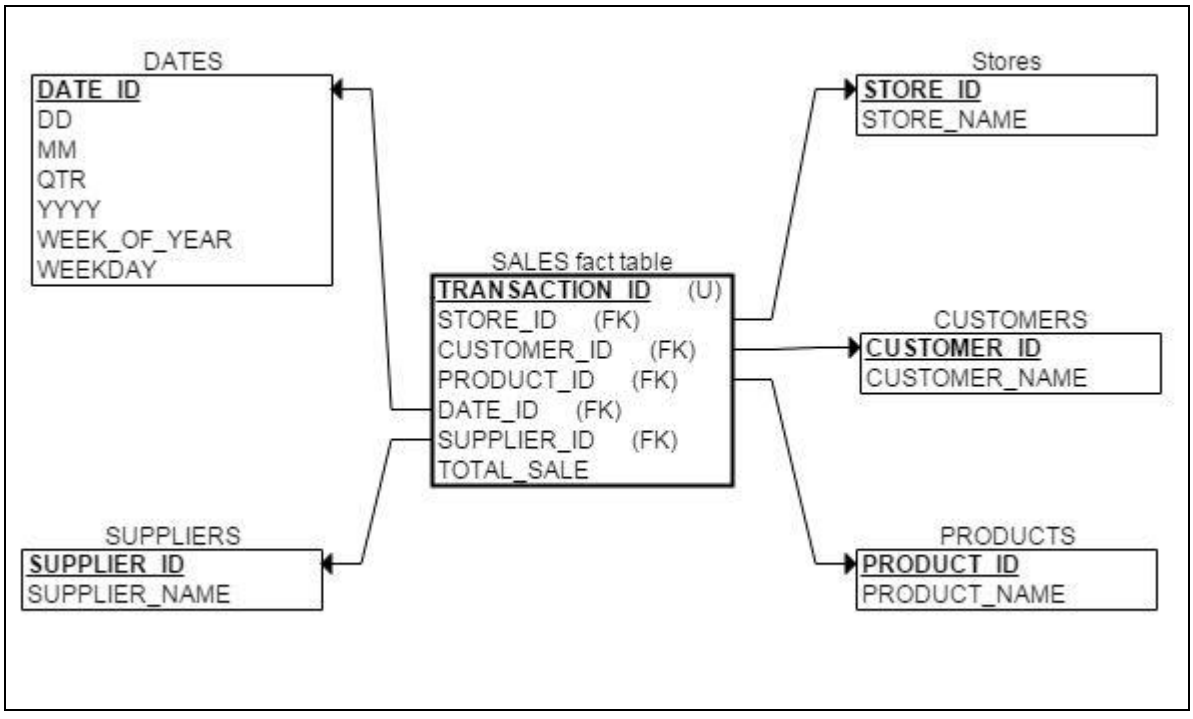
3 MESHJOIN Algorithm

Following is the diagram to show the implementation of Meshjoin algorithm in this project. All the 8 tasks are done by the above mentioned “Meshjoin Worker Thread”. The program dynamically loads the partition size and the stream partition’s records size from the global configuration file: system.properties.

One point hidden in the following diagram is: when all the data is loaded into the global Input Queue, and Meshjoin worker assembles them in partition and enqueue them in the stream queue, a series of dummy partitions are enqueued at the end of execution to make sure that every partition is processed and dequeued in the end.



4 Star Schema for Countdown Data Warehouse



5 OLAP Queries and Output

All the following SQL statements are based on Derby database system. Oracle SQL syntax is a little different with support of limited row number. Both versions of SQL files are provided under the DDL directory.

Derby:

```
SELECT * from sales FETCH FIRST 3 ROWS ONLY;
```

Oracle:

```
SELECT * from (
        SELECT * from sales
) where ROWNUM <= 3;
```

5.1 Which product generated maximum sales in Dec, 2014?

```
SELECT p.product_name, SUM (s.TOTAL_SALE) as product_total_sale from sales as s,products as p,dates as d
   WHERE d.mm=12 and
      d.DATE_ID = s.DATE_ID and
      p.PRODUCT_ID = s.PRODUCT_ID
      GROUP by p.product_name
      ORDER by product_total_sale desc
      FETCH FIRST 3 ROWS ONLY;
PRODUCT_NAME              |PRODUCT_TOT&
-------------------------------------------
Bouillon cubes            |1759.58
Kiwis                     |1757.75
Mac and cheese            |1632.00

3 rows selected
```

5.2 Which store produced highest sales in the whole year?

```
SELECT s.STORE_NAME, SUM (sales.TOTAL_SALE) store_sales from stores s, sales, dates d where
   s.STORE_ID = sales.STORE_ID and    sales.DATE_ID = d.DATE_ID and    d.YYYY = 2014
   GROUP by s.STORE_NAME
   FETCH FIRST 3 ROWS ONLY;
STORE_NAME        |STORE_SALES
---------------------------------
Albany            |80559.10
East Auckland     |78218.98
Henderson         |42562.12
```

5.3 Determine the supplier name for the most popular product based on sales

```
SELECT s.SUPPLIER_NAME, SUM (sales.TOTAL_SALE) supplier_sales from suppliers s, sales, products p where
   sales.SUPPLIER_ID = s.SUPPLIER_ID and
   p.PRODUCT_ID = sales.PRODUCT_ID
  GROUP by SUPPLIER_NAME
   FETCH FIRST 3 ROWS ONLY;
SUPPLIER_NAME              |SUPPLIER_SA&
--------------------------------------------
3Com Corp                  |40192.94
3M Company                 |41627.03
A.G. Edwards Inc.          |94744.25
```

5.4 Presents the quarterly sales analysis for all stores using drill down query concepts

```
SELECT stores.store_name, dates.qtr quarter, SUM (sales.total_sale) store_quarter_sales from stores, sales, dates
  WHERE stores.store_id = sales.store_id  AND sales.date_id = dates.date_id
  GROUP by stores.store_name, dates.qtr
  ORDER by store_name, qtr;
STORE_NAME        |QUARTER    |STORE_QUART&
--------------------------------------------
Albany            |1          |20775.95
Albany            |2          |18754.30
Albany            |3          |19518.13
Albany            |4          |21510.72
East Auckland     |1          |18717.08
East Auckland     |2          |19440.58
East Auckland     |3          |20262.88
East Auckland     |4          |19798.44
……OMITTED 40 – 16 = 24 RECORDS HERE……
Westgate          |1          |20248.07
Westgate          |2          |22615.39
Westgate          |3          |21495.99
Westgate          |4          |18410.83
Whangaparaora     |1          |20646.29
Whangaparaora     |2          |19896.91
Whangaparaora     |3          |22637.11
Whangaparaora     |4          |17377.91

40 rows selected
```

5.5 Create a materialised view with name "STOREANALYSIS" that present the product-wise sales analysis for each store.

- The following is the SQL and query result for product-wise sales for each store based on Derby database.

```
SELECT stores.store_name, products.product_name, SUM (sales.total_sale) store_quarter_sales from stores, sales,
products where
  stores.store_id = sales.store_id and
  sales.product_id = products.product_id
  GROUP by stores.store_name, products.product_name
  ORDER by store_name, product_name
  ;
STORE_NAME          |PRODUCT_NAME                |STORE_QUART&
-----------------------------------------------------------------
Albany              |Apples                      |581.44
Albany              |Applesauce                  |1688.10
Albany              |Asparagus                   |527.25
Albany              |Avocados                    |717.64
Albany              |BBQ sauce                   |675.84
Albany              |Bagels                      |402.93
……OMITTED MANY RECORDS HERE……
Whangaparaora       |Tuna / Chicken              |734.40
Whangaparaora       |Vegetable oil               |255.56
Whangaparaora       |Vegetables                  |1074.24
Whangaparaora       |Veggie burgers              |223.80
Whangaparaora       |Veggies                     |411.33
Whangaparaora       |Vinegar                     |1089.00
Whangaparaora       |Worcestershire sauce        |321.86

987 rows selected
```

- Materialized View is supported by Oracle database system, here is the SQL output for creating the materialized view:

```
CREATE materialized view STOREANALYSIS as
  SELECT stores.store_name, products.product_name, SUM (sales.total_sale) store_quarter_sales from stores, sales,
products where
  stores.store_id = sales.store_id and
  sales.product_id = products.product_id
  GROUP by stores.store_name, products.product_name
  ORDER by store_name, product_name;

materialized view STOREANALYSIS created.
```

6 Lessons Learned

- Writing to database is far slower than reading from database. The Data Loader Thread is the first one started, yet it is the last to finish. If a realtime ETL program is to be implemented in reality, the writing data must become a bottleneck of the system, which must be delibrately designed. A further investigation on this issue need more study.

```
INFO: Realtime Data Producer Completed! Runtime: 1 seconds
INFO: Meshjoin Worker Thread Runtime: 1 seconds
INFO: Data Loader Thread Runtime: 9 seconds
```

- Database systems are compatible only in a sub set of SQL. Migrating from one DBMS to another may be a pain for a database administrator. Using Java based locally running DBMS Derby is very helpful for development and test at first on personal laptop. When testing and running the application on Oracle database, the SQL statements have to be written again.
- A sequence of single java.sql.Statement for insersion of a record is too slow. Using java.sql.PreparedStatement with batch support have greatly improved the overall performance.
- The records size in each dimension tables is relatively small, caching them in memory is feasible and could contribute the performance improvement.