Deploying an IBM Industry Data Model on an IBM Netezza data warehouse appliance

Whitepaper

Contents

Who Should Read This Document

- IT Architects
- IT Specialists
- Data Modelers
- Database Administrators
- Business Intelligence practitioners

About This Paper

This paper provides guidance on how to deploy an IBM Industry Data Model on an IBM Netezza data warehouse appliance. This guidance is based on practical experience gained from a sample deployment which was performed by the IBM Industry Models development team and the IBM Netezza Product Integration team.This paper is divided into the following sections:

The **Introduction** section provides an overview of both the IBM Industry Models and IBM Netezza data warehouse appliances. This section also describes the approach used when implementing projects using IBM Industry Models and describes the key steps required to deploy an Industry Model on an IBM Netezza data warehouse appliance.

Chapter 1, "**Transforming the Logical Data Model to a Physical Data Model**" outlines the steps required to transform an IBM Industry Data Model to a physical data model for an IBM Netezza data warehouse appliance using IBM InfoSphere Data Architect (IDA).

Chapter 2, "**Defining a partitioning strategy in order to distribute the data**" describes the process for distributing data in a way that influences the performance of the IBM Netezza data warehouse appliance.

Chapter 3, "**Generating & executing DDL in order to create a IBM Netezza data warehouse appliance database**" describes the process for generating and executing the DDL to create a database on an IBM Netezza data warehouse appliance

Chapter 4, "**Performance Tuning & Query Response Times**" describes the performance tuning steps executed on the IBM Netezza data warehouse appliance once the sample data was analyzed, and provides details of the query responses times.

Appendix, "**Details of the Queries Used in the Performance Testing**" provides a full list and details of the sixteen queries used to test the database before and after performance tuning.

Introduction

IBM Industry Models and Netezza combined can address the ever increasing demands placed on the data warehouse

In an environment where customers are under pressure to create solutions more quickly, to respond to external regulations, develop competitive advantage and cope with reducing budgets - speed of delivery and appropriateness of solution design are increasingly important requirements. Key to success is a high-performing and reliable database management system, and crucial to the long-term goals of the customer is an industry specific, scalable business data model. Together, these components can assist with the rapid deployment of focused customer solutions, while also providing the opportunity to refine, grow and mirror the real business requirements of the organization. IBM Industry Data Models and IBM Netezza data warehouse appliances are a strong partnership which can address the business and technical database needs of the customer.

A solution for data warehousing that includes IBM Industry Data Models and the IBM Netezza data warehouse appliance combines the industry knowledge, power and performance necessary to keep up with expanding business requirements.

The Components

IBM Netezza

IBM Netezza data warehouse appliances provide an easily maintained database management system that leverages high-performance data optimization analytics

IBM Netezza is a family of data warehouse appliances and software products that allow you to easily deploy high-performance data analytics across your entire enterprise. IBM Netezza appliances integrate database, server and storage into a single, easy to manage system that requires minimal set-up and administration. The IBM Netezza data warehouse appliance helps companies analyze rapidly growing data volumes very quickly for faster speed-to-market and better decision-making. With simple deployment, out-of-the-box optimization, no tuning and minimal on-going maintenance, the IBM Netezza data warehouse appliance has the industry's fastest time-to-value and lowest total-cost-of-ownership.

IBM Industry Data Models are a set of integrated data design models with a strong business perspective

Customers are able to easily and cost-effectively scale their business intelligence (BI) and analytical infrastructure, leveraging deeper insights from growing data volumes throughout the organization.

With rapid results on the most complex queries, low maintenance and fast setup, the IBM Netezza data warehouse appliance gives organizations a reliable and valuable source of critical information.

IBM Industry Data Models

IBM Industry Data Models combine deep expertise and industry best practice in a usable form for both business and IT.

IBM Industry Data Models complement the information architecture by providing design level models for the data warehouse and data mart structures on Netezza

IBM Industry Data Models are a family of data models designed to accelerate enterprise data warehouse business intelligence solutions. They contain multiple design elements which can be used to create a range of data warehouse solutions from departmental data marts to enterprise-wide data warehouses and have business elements specific to the customers industry including Banking, Financial Markets, Insurance Retail, Healthcare and Telecommunications.

The Business Data Model documents the enterprise business terms and project scope

A key strength of IBM Industry Data Models is the ability to derive focused data-warehouse solutions by recognizing and incorporating the project's business goals early in the development cycle. To do this, each set of industry models comes with three core models that are fully integrated to provide an end-to-end design solution.

The Atomic Warehouse Design Model is the design for a consolidated enterprise data hub

- **Business Data Model** – a vocabulary of the business terms used throughout the data architecture. The terms can be grouped into areas of interest to identify the requirements (measures and dimensions) for a piece of analysis or downstream data-mart. This model is fully integrated with companion models so it can provide full traceability of business requirements defined in the vocabulary to the resulting supporting technical components. It can also be used to scope the underlying technical components for a particular business issue thereby ensuring a focused data warehouse or data mart solution. This can simplify the process of identifying source data requirements and can reduce the implementation timeline.

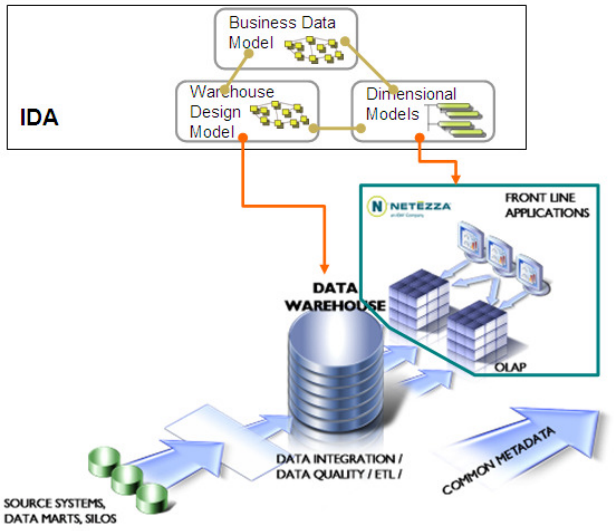
The Dimensional Warehouse Model is the design for dimensional data structures optimized for queries and analytics

- **Atomic Warehouse Model** - a logical entity-relationship design model for an enterprise data warehouse. This model specifies the design structures for consolidating and storing

the history of normalized operational data from across the entire enterprise. It acts as a data integration hub from which analytical reporting requirements and data-marts can obtain reliable data. It is a fully scalable model that can be built in phases determined by the business issues identified in the Business Data Model.

- **Dimensional Warehouse Model** – The Dimensional Warehouse Model is a logical model derived from the Business Data Model and is an optimized data repository for supporting analytical queries. It is a set of pre-defined logical star schema data marts, consisting of fact entities and conformed dimensions.

  The Dimensional Warehouse Model can be integrated with the Atomic Warehouse Model or it can be deployed separately as a fully independent dimensional star-schema model on an IBM Netezza data warehouse appliance.



In a typical customer engagement, the initial step is the capturing of business reporting requirements using the Business Data Model. This is then used to determine the scope and customization requirements of the underlying Atomic Warehouse Model and Dimensional Warehouse Model, ensuring a business focused and targeted data solution is created. Alternatively, the pre-defined set of data marts that capture some of the key business solutions can be deployed directly. These data marts -- defined as logical data models in IBM InfoSphere Data Architect (IDA) -- can be very easily transformed to IBM Netezza specific DDL.

This whitepaper describes a sample deployment of a subset of an IBM Industry Data Model on an IBM Netezza data warehouse appliance. A pre-defined star schema provided in an IBM Industry Data Model was transformed to a physical data model, the corresponding DDL was generated and deployed on an IBM Netezza data warehouse appliance. Over 180 million rows of data were loaded and analyzed. Based on the data characteristics, the database was tuned to maximize query response times resulting in an 8-fold performance improvement.

This whitepaper uses the “Item Sales Transaction Fact” star schema as an example and deploys it on an IBM Netezza data warehouse appliance.

The steps performed are as follows:

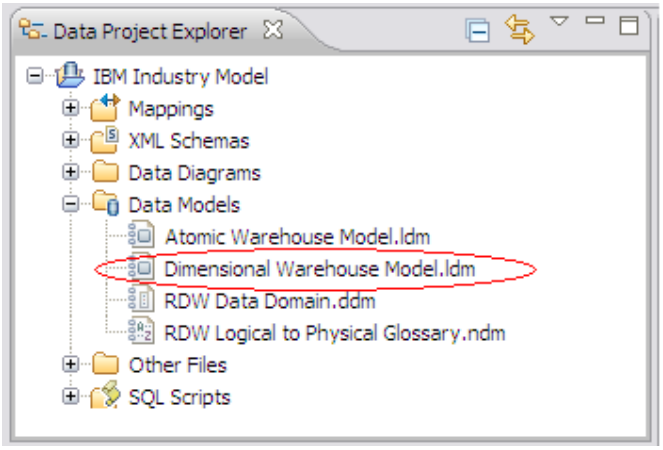
1. Examine the pre-defined data mart in IDA and transform the logical data model to an IBM Netezza physical data model
2. Define a partitioning strategy for distributing the data
3. Generate a DDL script for IBM Netezza and execute the DDL script on the IBM Netezza data warehouse appliance
4. Execute query performance testing and tuning on IBM Netezza

Chapter 1: Transforming the Logical Data Model to a Physical Data Model

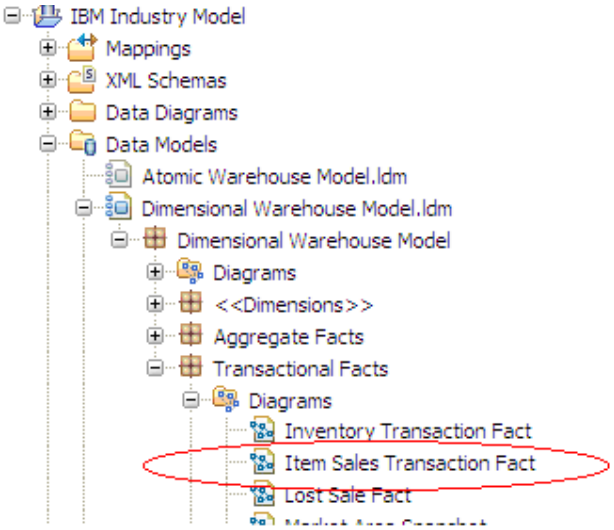
To begin the deployment process, the logical data mart model must be transformed to a physical data model – a model which takes into account the facilities and constraints of a given database management system (e.g. an IBM Netezza data warehouse appliance)

IBM InfoSphere Data Architect (IDA) – has the ability to transform a logical data model into an IBM Netezza physical data model, using the following procedure:

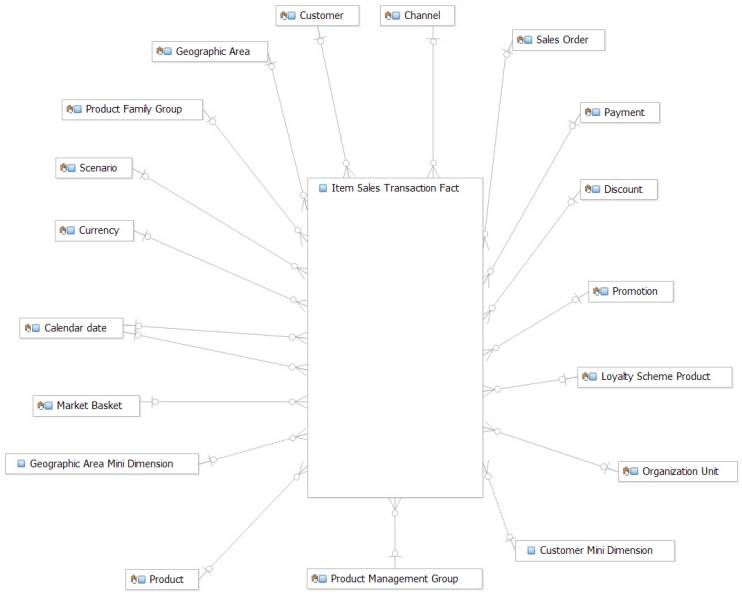
Open the Dimensional Warehouse Model in IDA.



Choose the diagram or data mart to be transformed from logical to physical. In the sample deployment, the “Item Sales Transaction Fact” star-schema diagram was chosen.



Open the diagram to view the star-schema represented in entity-relationship diagram format



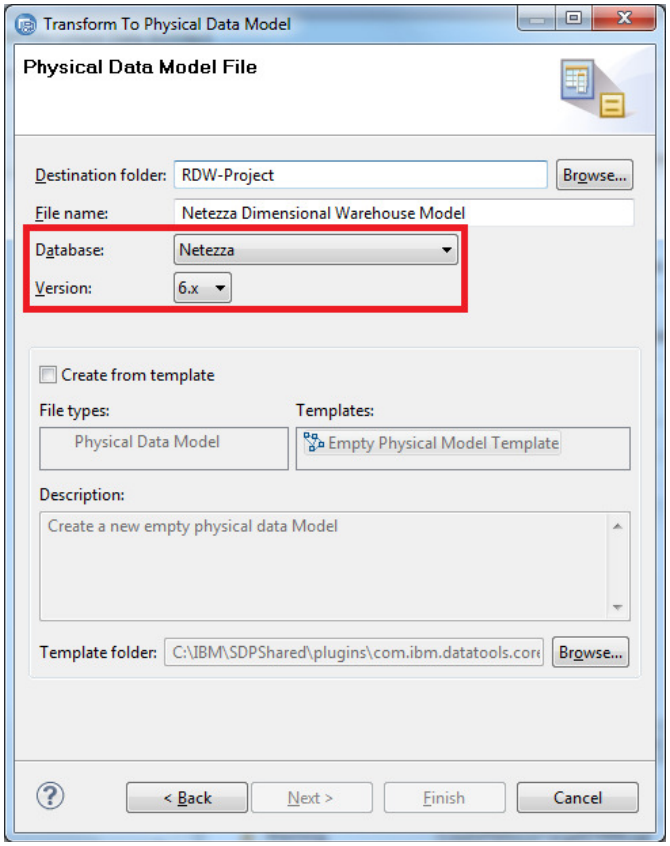
In a full project, some customization of the logical models would occur at this point, to refine the models for the specific business reporting or analytical requirements. In the sample deployment, an out of the box data mart from the Dimensional Warehouse Model is chosen and transformed to an IBM Netezza physical model without any customization.

Open the diagram, then select the menu option: Data > Transform > Physical Data Model

In the resulting wizard, choose the following options:

Database: Netezza

Version: 6.x



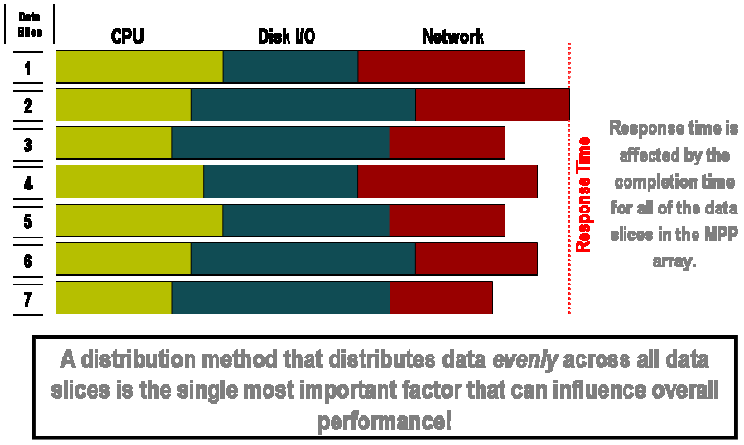
Click Next followed by Finish to complete the creation of the Physical Data Model.

Chapter 2: Defining a partitioning strategy in order to distribute the data

IBM Netezza data warehouse appliances dramatically simplify the performance tuning requirements of the RDBMS

One of the strengths of an IBM Netezza data warehouse appliance is that performance tuning is dramatically simplified. Many of the steps other RDBMSs require for configuration and tuning – for example: tablespace configuration, physical & logical log sizing, block sizing, extent sizing, temp space allocation, etc. are not required using IBM Netezza data warehouse appliances.

A distribution method that distributes data evenly across all data slices is the single most important factor that can influence the overall performance of the IBM Netezza data warehouse appliance. Good distribution is a fundamental element of performance.



IBM Netezza data warehouse appliances benefit from a simple partitioning strategy for distributing data

IBM Netezza data warehouse appliances benefit from a simple partitioning strategy for distributing data, which includes three options:

1. DISTRIBUTE ON RANDOM: Use a Random distribution – this will distributed data randomly, resulting in even distribution across the data slices. Random distribution may result in significant data movement during queries which can slow query performance.
2. DISTRIBUTE ON (COL1..COL4): Explicitly define a distribution key in the Physical Data Model – although up to four columns is supported, the recommendation is to use one column for the distribution key. Columns used as join keys are optimal. This distribution method is recommended for frequently used large tables and has optimal performance.

3. If you don't specify a distribution key one will be selected by the system, typically the first column. This may not be the optimal choice, so generally this option is not recommended.

When choosing a distribution key consider the following factors:

- The more distinct the distribution key values, the better
- Tables used together should use the same columns for their distribution key where possible
- If a particular key is largely used in equi-join clauses, then that key is a good choice for the distribution key
- Check that there is no accidental processing skew[1]. Good record distribution avoids processing skew, optimizing parallel processing.

Random distribution is often the best choice to guarantee even distribution of the data. In some cases – for example, when co-location of data is desired – the random distribution can be overridden by explicitly defining a distribution key in the physical data model.

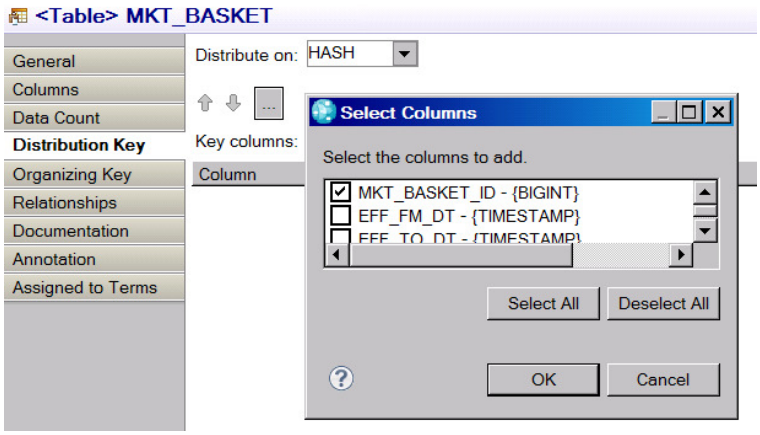
IBM InfoSphere Data Architect supports definition of distribution keys - either random or explicitly defined keys - in the physical data model

In this sample deployment, the smaller dimension tables were partitioned on a random distribution. The larger MKT_BASKET dimension table was partitioned on MKT_BASKET_ID to ensure co-location for joins between the Fact table and this large, frequently used dimension. The fact table ITM_SALE_TXN_FCT was therefore also partitioned on the same distribution key. This explicit distribution setting prevents redistribution of data for joins between the fact and the market basket table.

To explicitly define a distribution key in the IDA Physical Data Model:

- Select a table
- Select Properties > Distribution Key.
- Distribute on: choose HASH
- Use the ellipse button to add one or more distribution columns. Up to four columns can be used to define the distribution key.

[1] Skew is the difference in processing time that results when more data is being processed by one or a few nodes when compared with all nodes. A typical example: joins on 'gender' would use only two of the systems nodes for the join.



Column(s) used for distribution keys should be integer (any size) or temporal (date or time) types. String, double, float and other data types are not recommended as distribution keys. Distribution keys should have high cardinality to help avoid data skew.

Always review the distribution of data after implementation to ensure data is distributed evenly and that data skew has been minimized.

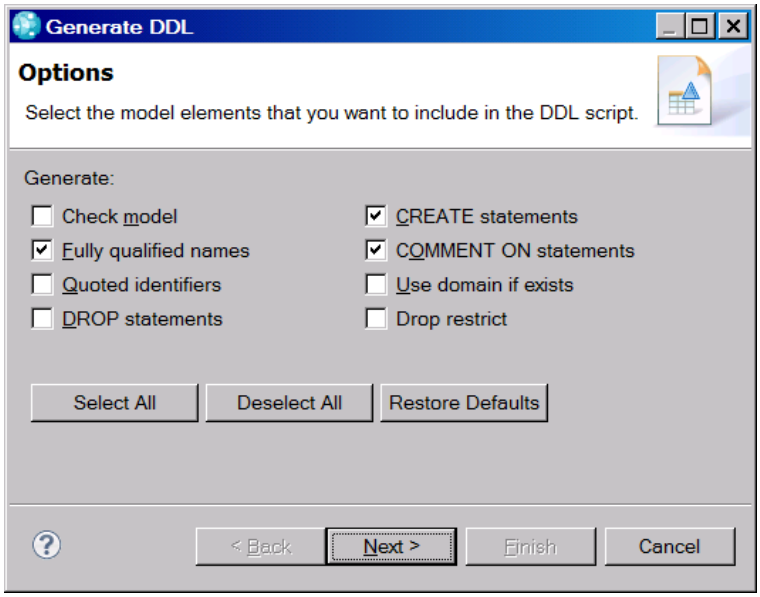
Chapter 3: Generating & Executing DDL in order to create an IBM Netezza data warehouse appliance database

IBM InfoSphere Data Architect supports the generation of IBM Netezza specific DDL from the physical data model

Now that the physical data model structure is defined, the next step is to generate the DDL instructions for the creation of the database on the IBM Netezza data warehouse appliance. This is achieved using the DDL generation capability provided by IDA.

Generating the DDL

Select the Physical Data Model and click on Data > Generate DDL...

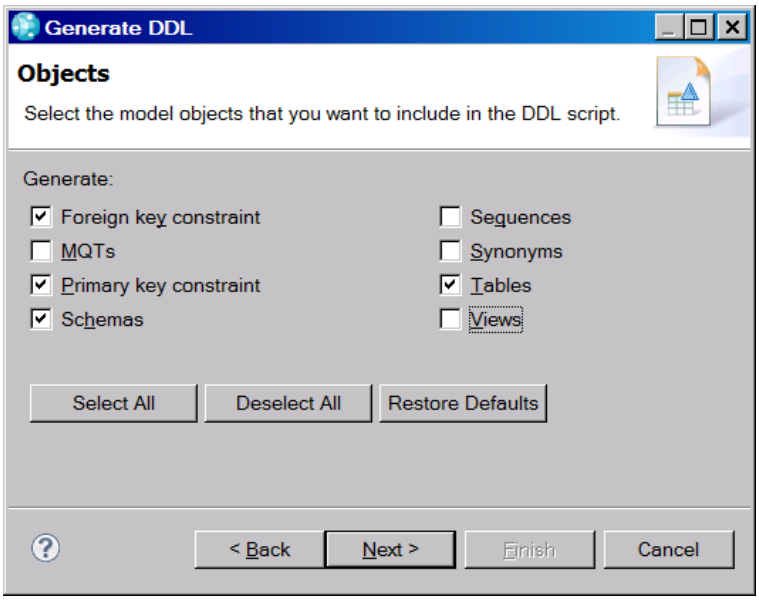


Follow the steps in the wizard to create a DDL script and save it in your IDA project.

Depending on the preferences chosen, the DDL script can contain:

- Create Table statements with specification of Distribution Key
- Primary Key constraints[2]
- Foreign Key constraints[2]
- Table & column documentation/comments

[2] Note: defining constraints is strongly recommended as it provides useful information to the optimizer and BI tools



Execute the DDL to create the database structures on the IBM Netezza data warehouse appliance

Establishing a connection from IDA to the IBM Netezza data warehouse appliance

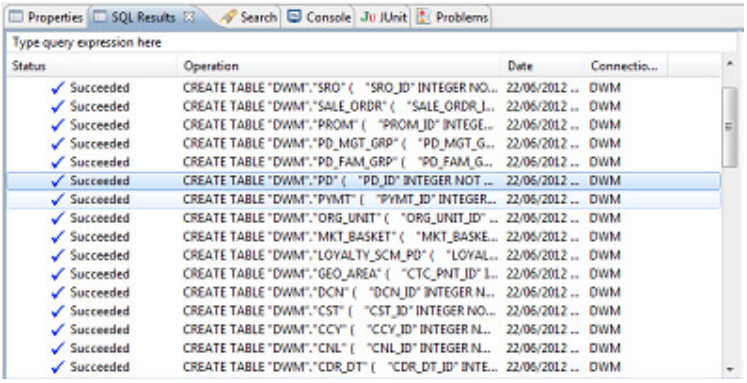
In the Data Source Explorer window, right-click on Database Connections and choose New. Select Netezza as the database manager, and Netezza 6x JDBC Driver Default. Fill in the details for the target IBM Netezza system and test connection.

Executing the DDL to create the IBM Netezza database

Finally, the DDL script that has been generated from the IBM Industry Data Model can be executed to create the relational database structures.

Select Run > Run SQL

The results of the DDL execution can be viewed in the SQL Results panel.



Chapter 4: Performance Tuning & Query Response Times

Over 180 million rows of sample data were imported during the sample deployment in order to perform query analysis

The subset of the IBM Industry Data Model used in this demonstration was populated with sample data counts as follows:

Logical Entity	Physical Table	Row count
Item Sale Transaction Fact	ITM_SALE_TXN_FCT	184,260,608
Calendar Date	CDR_DT	3,654
Customer	CST	457,105
Customer	CST_MINI_DIMENSION	228,553
Geographic Area	GEO_AREA	176
Geographic Area	GEO_AREA_MINI_DIMENSION	88
Market Basket	MKT_BASKET	76,404,896
Organisation Unit (Store)	ORG_UNIT	176
Product	PD	51,842
Product Family Group	PD_FAM_GRP	10
Product Management Group	PD_MGT_GRP	1,304
Payment	PYMT	76,404,896

A number of other dimensions, although included in the sample deployment, were populated only with minimal data to support the joins.

Performance Tuning

In order to maximize performance on the IBM Netezza data warehouse appliance, the following alterations were made:

Effective performance tuning includes distribution of the data, automatic zone maps and data re-organization

1. Co-location of joins

As the sample data size and distribution was known, this step was completed in the IDA tooling while specifying the partitioning strategy for the database tables (see chapter 2).

2. Zone Maps

Zone Maps are an automatic IBM Netezza data warehouse appliance function for segmenting columns of data based on the values in that data.
Columns of data are broken down into blocks where the IBM Netezza data warehouse appliance stores only the first and last record in each block e.g. minimum and maximum values. This allows for faster query access by matching the data requirement to the block and only reading the relevant blocks of interest.

To maximize Zone Map potential: organize the fact table so the most used dimension keys (in this case, CDR_DT_ID, CST_ID, MKT_BASKET_ID, PD_ID) make use of the IBM Netezza data warehouse appliance zonemaps.

- Any filter/predicate on the dimensions that results in filtering on these keys, will be pushed down as bitmap filters on the fact table (volume restrictions apply).

3. Re-organization of the data

Run the groom command to cluster/organize the rows in the fact table for optimized filters/predicates on the above mentioned columns.

Finally, update statistics on all tables.

Query Response Time

A set of 16 sample queries, based on real-life analysis requirements, was developed to assess query response time

Sample queries and response time results

A set of sample queries was prepared, based on real-life analysis requirements of the data. The detailed queries, including SQL, are included in the Appendix.

Query#	Description
1	Sales & profit over time, grouped by year
2	Drill down of query #1 by month
3	Drill down of query #1 by week
4	Sales & profit by Product Group for a specific time period
5	Drill down of query #4 by product group
6	Drill down of query #4 by product group and product
7	Sales & profit by Geographic Location for a specific time period
8	Drill down of query #7 by location (country)
9	Drill down of query #7 by location (country and state)
10	Sales & profit by Store for a specific time period (month)
11	Drill down of query #10 by country and state
12	Time-based year-to-year comparative analysis
13	Time-based year-to-year comparative analysis by product family
14	Data mining type query: Average Market Basket Values grouped by quarter
15	Data mining type query: Average Market Basket Values grouped by quarter
16	Data mining type query: Average Market Basket Values ranked by count, total & average

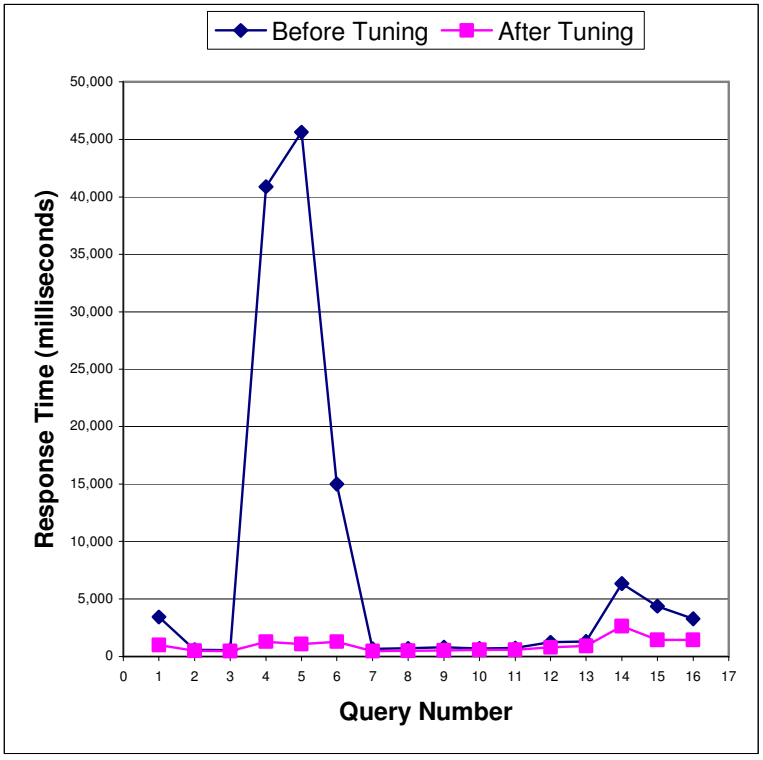
The set of queries was run using Apache's JMeter tool to simulate concurrent usage and measure the response time.

After simple performance tuning, an average query response improvement of 8 times the pre-tuning performance was recorded

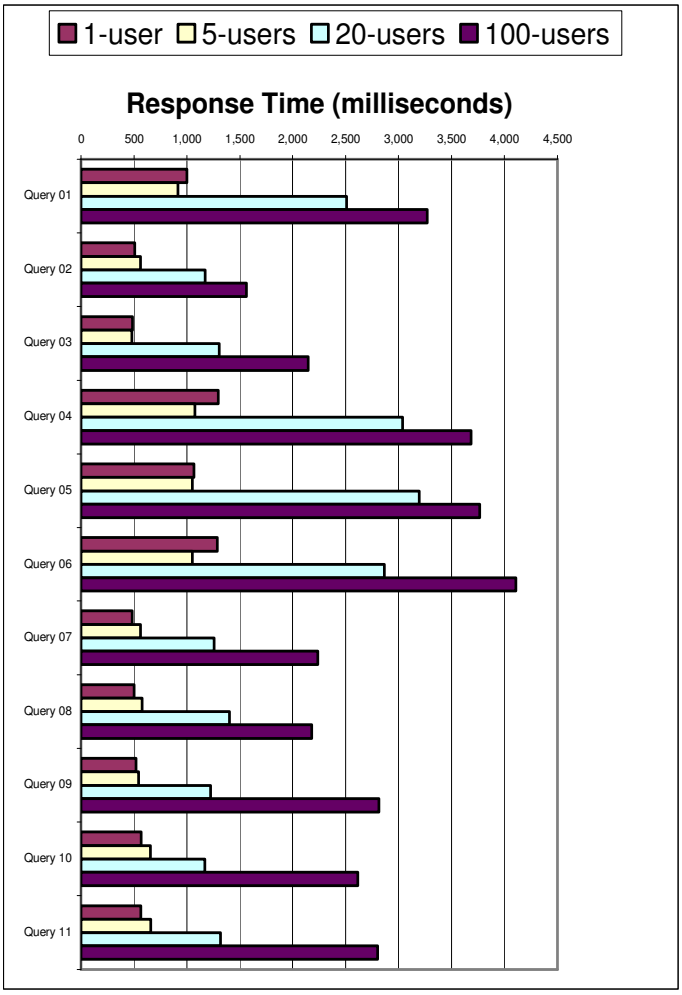
The table below lists the average time (in milliseconds) to run each query before and after simple performance tuning.

Query #	Before	After	Improvement
1	3,443	1,001	344%
2	567	507	112%
3	555	485	114%
4	40,874	1,294	3159%
5	45,620	1,068	4272%
6	15,014	1,290	1164%
7	654	482	136%
8	704	503	140%
9	799	520	154%
10	689	570	121%
11	730	567	129%
12	1,222	788	155%
13	1,309	915	143%
14	6,336	2,638	240%
15	4,376	1,441	304%
16	3,281	1,442	227%
Total	126,170	15,508	814%

The response time & improvement is also illustrated in the following chart:



In addition, queries 1 to 11 -- which are typical examples of the types of queries that would be run concurrently by different users in a real-life scenario -- were further analysed to assess performance when executed concurrently. The chart below lists the average query response time when queries were executed simultaneously with concurrent usage of 1, 5, 20 & 100 concurrent users.



Conclusion

IBM Industry Data Models are well suited design models for IBM Netezza data warehouse appliance implementations. IBM Industry Data Models enhance the IBM Netezza data warehouse appliance with industry specific business content which facilitates downstream query and reporting analysis.

IBM Netezza data warehouse appliances also have the facility to leverage the IBM Industry Model Dimensional Warehouse Models, storing information in a typical star-schema, with the ability to combine and aggregate the data for more complex queries. This sample implementation demonstrated how a pre-defined IBM Industry Data Model star-schema was quickly & easily deployed on an IBM Netezza data warehouse appliance and how performance tuning for the data set resulted in excellent query response times (8 times improvement).

The combination of IBM Industry Data Models and IBM Netezza data warehouse appliance provides a number of benefits:

Total Coverage - for the deployment of analytics from business definition to appliance setup. IBM Industry Data Models provide comprehensive business content for multiple industries.

Flexibility - to drive out the subset of this content onto IBM Netezza – enables rapid deployment of analytics specific to business issues of the bank

Scalability - highly scalable solution in terms of both business function scalability and performance/throughput scalability

Proven – IBM Industry Models have been used by over 500 global clients and the business content has been pre-tested for deployment on IBM Netezza data warehouse appliances.

By combining IBM Industry Data Model capabilities with the industry-leading performance of IBM Netezza data warehouse appliances, organizations can fully leverage the value of information currently trapped in different data silos to drive innovation and improve business results.

Appendix: Details of the queries used in the performance testing

Query 1:

```
-- Sales & profit over Time
-- Group measures by year
select
dt.cdr_yr,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.CDR_DT dt
Where fact.CDR_DT_ID = dt.CDR_DT_ID
group by dt.cdr_yr
order by dt.cdr_yr
```

Query 2:

```
-- As 1, with drill down on year listing months
select
dt.cdr_yr,
dt.cdr_mo,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
dt.CDR_YR = 2008
group by dt.cdr_yr, dt.cdr_mo
order by dt.cdr_yr, dt.cdr_mo
```

Query 3:

```
- As 1, with drill down on a year listing weeks
Select dt.cdr_yr, dt.cdr_mo, dt.cdr_wk_num,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
dt.CDR_YR = 2008
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_wk_num
order by dt.cdr_yr, dt.cdr_mo
```

Query 4:

```
-- Sales & Profit by Product Group
-- Group measures by Highest tier of "Product Hierarchy"
Select   level_2.GRP_NM, sum(fact.NUM_OF_ITM_SOLD) as
NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact
left outer join
  DWM.PD_MGT_GRP grp on fact.PD_MGT_GRP_ID =
grp.PD_MGT_GRP_ID
left outer join
  DWM.PD_MGT_GRP level_1 on grp.PRN_PD_GRP_ID = lev-
el_1.PD_MGT_GRP_ID
left outer join
  DWM.PD_MGT_GRP level_2 on level_1.PD_MGT_GRP_ID =
level_2.PD_MGT_GRP_ID
group by   level_2.GRP_NM
order by SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 5:

```
-- Drill down through product group
Select level_2.GRP_NM,   grp.GRP_NM,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from   DWM.ITM_SALE_TXN_FCT fact
left outer join   DWM.PD_MGT_GRP grp on fact.PD_MGT_GRP_ID
= grp.PD_MGT_GRP_ID
left outer join   DWM.PD_MGT_GRP level_1 on
grp.PRN_PD_GRP_ID = level_1.PD_MGT_GRP_ID
left outer join   DWM.PD_MGT_GRP level_2 on lev-
el_1.PD_MGT_GRP_ID = level_2.PD_MGT_GRP_ID
where level_2.GRP_NM like 'Entertainment%'
group by   level_2.GRP_NM, grp.GRP_NM
order by 2, SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 6:

```
-- Drill down through product group hierarchy all the way to individ-
ual products
Select   level_2.GRP_NM,   level_1.GRP_NM, grp.GRP_NM,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from   DWM.ITM_SALE_TXN_FCT fact
left outer join
  DWM.PD_MGT_GRP grp on fact.PD_MGT_GRP_ID =
grp.PD_MGT_GRP_ID
left outer join
  DWM.PD_MGT_GRP level_1 on grp.PRN_PD_GRP_ID = lev-
el_1.PD_MGT_GRP_ID
left outer join
  DWM.PD_MGT_GRP level_2 on level_1.PRN_PD_GRP_ID =
level_2.PD_MGT_GRP_ID
```

```
Where level_2.GRP_NM like 'Food%' and
level_1.GRP_NM like 'Lunchtime%'
group by   level_2.GRP_NM,   level_1.GRP_NM, grp.GRP_NM
order by 1, 2, 3, SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 7:

```
-- Sales & Profit by Geographic Location
-- Group measures by Highest tier of "Geographic Area"
select
dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.CTY,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from DWM.ITM_SALE_TXN_FCT fact, DWM.GEO_AREA geo,
DWM.CDR_DT dt
Where fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.GEO_AREA_ID = geo.CTC_PNT_ID and
dt.cdr_yr = 2008 and dt.cdr_mo = 4
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.CTY
order by dt.cdr_dt, SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 8:

```
-- Drill down through product location hierarchy all the way to indi-
vidual locations
Select dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.RGON,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.GEO_AREA geo,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.GEO_AREA_ID = geo.CTC_PNT_ID and
dt.cdr_yr = 2008 and dt.cdr_mo = 4 and geo.CTY like 'USA%'
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.RGON
order by dt.cdr_dt, SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 9:

```
-- Drill down through product location hierarchy all the way to indi-
vidual locations
Select dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.CITY,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.GEO_AREA geo,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.GEO_AREA_ID = geo.CTC_PNT_ID and
dt.cdr_yr = 2008 and dt.cdr_mo = 4 and geo.CTY like 'USA%' and
geo.RGON like 'CA%'
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, geo.CITY
```

```
order by dt.cdr_dt, SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 10:

```
-- Sales & Profit by Store
-- Group measures by Highest tier of "Organization Unit"
select
dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, org.DSC,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.ORG_UNIT org,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.ORG_UNIT_ID = org.ORG_UNIT_ID and
dt.cdr_yr = 2008 and dt.cdr_mo = 4
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, org.DSC
order by dt.cdr_dt,  SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 11:

```
-- Drill down through Organization Unit hierarchy all the way to indi-
vidual stores
-- Organization Unit Locations may be mixed with Geographic Loca-
tions - eg stores by country / state
Select dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, org.DSC,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.ORG_UNIT org,
  DWM.GEO_AREA geo,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.ORG_UNIT_ID = org.ORG_UNIT_ID and
fact.GEO_AREA_ID = geo.CTC_PNT_ID and
geo.CTY like 'USA%' and geo.RGON like 'CA%' and
dt.cdr_yr = 2008 and dt.cdr_mo = 4
group by dt.cdr_yr, dt.cdr_mo, dt.cdr_dt, org.DSC
order by dt.cdr_dt,  SALE_AMT_EXCL_OF_SALE_TAX desc
```

Query 12:

```
-- Comparative analysis - year to year, based on specific time peri-
od. Comparative analysis will usually extract both sets and display
side by side with calculated increase / decrease values and per-
centage change.

Select dt.cdr_mo, dt.cdr_yr,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
```

```
  DWM.CDR_DT dt
Where fact.CDR_DT_ID = dt.CDR_DT_ID   and
dt.cdr_dt between '2007-01-01' and '2007-05-12'
group by dt.cdr_mo, dt.cdr_yr
union all
select dt.cdr_mo, dt.cdr_yr,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.CDR_DT dt
Where fact.CDR_DT_ID = dt.CDR_DT_ID   and
dt.cdr_dt between '2008-01-01' and '2008-05-12'
group by dt.cdr_mo, dt.cdr_yr
order by 2, 1
```

Query 13:

```
-- Similar to 12, split by product family group
select
dt.cdr_mo, dt.cdr_yr,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.PD_FAM_GRP fam,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.PD_FAM_GRP_ID = fam.PD_FAM_GRP_ID and
dt.cdr_dt between '2007-01-01' and '2007-05-12' and
fam.BRND like 'Food%'
group by dt.cdr_mo, dt.cdr_yr
union all
select dt.cdr_mo, dt.cdr_yr,
sum(fact.NUM_OF_ITM_SOLD) as NUM_OF_ITM_SOLD,
sum(fact.SALE_AMT_EXCL_OF_SALE_TAX) as
SALE_AMT_EXCL_OF_SALE_TAX,
sum(fact.COST_OF_GOODS_SOLD_COGS) as
COST_OF_GOODS_SOLD_COGS
from
  DWM.ITM_SALE_TXN_FCT fact,
  DWM.PD_FAM_GRP fam,
  DWM.CDR_DT dt
where
fact.CDR_DT_ID = dt.CDR_DT_ID and
fact.PD_FAM_GRP_ID = fam.PD_FAM_GRP_ID and
dt.cdr_dt between '2008-01-01' and '2008-05-12' and
fam.BRND like 'Food%'
group by dt.cdr_mo, dt.cdr_yr
order by 2, 1
```

Query 14:

```
--Average Market Basket Values with and without Target Product
Select 'with product' as description,
count(distinct mkt_basket_id) as basket_count,
sum(sale_amt_incl_of_sale_tax) as basket_total,
basket_total / basket_count as basket_avg
from itm_sale_txn_fct fact
where mkt_basket_id in
(Select fact.mkt_basket_id
From itm_sale_txn_fct fact, pd pd
Where fact.pd_id = pd.pd_id
And pd.dsc = 'Baby Food-pd228')
union all
select 'without' as description,
count(distinct mkt_basket_id) as basket_count,
sum(sale_amt_incl_of_sale_tax) as basket_total,
basket_total / basket_count as basket_avg
from itm_sale_txn_fct fact
where mkt_basket_id not in
(select fact.mkt_basket_id
From itm_sale_txn_fct fact, pd pd
Where fact.pd_id = pd.pd_id
And pd.dsc = 'Baby Food-pd228')
```

Query 15:

```
--Average Market Basket Values by Quarter
Select dt.cdr_qtr,
count(distinct mkt_basket_id) as basket_count,
sum(sale_amt_incl_of_sale_tax) as basket_total,
basket_total / basket_count as basket_avg
from itm_sale_txn_fct fact, cdr_dt dt
where
fact.cdr_dt_id = dt.cdr_dt_id
group by dt.cdr_qtr
order by dt.cdr_qtr
```

Query 16:

```
--Average Market Basket Values by Store for a specific Product
Select org.dsc,
count(distinct mkt_basket_id) as basket_count,
sum(sale_amt_incl_of_sale_tax) as basket_total,
basket_total / basket_count as basket_avg,
rank() over(order by basket_count desc) rank_by_count,
rank() over(order by basket_total desc) rank_by_total,
rank() over(order by basket_avg desc) rank_by_avg
from itm_sale_txn_fct fact, org_unit org
where fact.org_unit_id = org.org_unit_id
and mkt_basket_id in
(Select fact.mkt_basket_id
From itm_sale_txn_fct fact, pd pd
Where fact.pd_id = pd.pd_id
And pd.dsc = 'Baby Food-pd228')
group by org.dsc
order by basket_avg desc
```