



Practical D1

Star and Snowflake Schemas

Objectives of this practical

- Create a simple data warehouse with a Star Schema/Snowflake Schema in Microsoft SQL Server
- Become familiar with the concepts what are Fact Tables and Dimension Tables in a Data Warehouse
- Understand how to connect Dimension tables to the Fact table in the data warehouse
- Write queries to extract useful information from the data

Section 1: MyStationerySupplies

MyStationerySupplies is a local business supplying a wide variety of stationery products to both retail and business customers. The company was founded in 2010 by Diploma in Business Information Technology (DBIT) student Amelia Koh.

When she first started her business, Amelia focused on a small clientele of regular home-based customers in Singapore who got to know about her business through word-of-mouth. Over time, her business started to grow and today, the company boasts of hundreds of monthly transactions and has regional offices based in Singapore, Malaysia, Indonesia and Thailand. From a humble one-man show in the earlier days, Amelia's company now employs over 30 staff.

The huge volume of sales transactions generated on a monthly basis has resulted in a constant struggle for the management of MyStationerySupplies to collate accurate and timely sales reports that can help them plan their business strategies.

With multiple offices located in diverse locations, each with their own database formats and transaction systems, the management of MyStationerySupplies cannot retrieve their sales data easily. Currently, sales data is collected manually by the different departments and groups for analysis in spreadsheets and the data from one source is often not consistent with one another. There are also complaints that the production database sometimes slow down to a halt when the executives run queries against it.

With the above situation experienced by MyStationerySupplies, it seems that the organization could benefit from a data warehouse. You are a Data Specialist who has been tasked to come up with a simple data warehousing solution that will help MyStationerySupplies's management solve their data analytics problems so as to help them make better business decisions.

Section2: Star Schema for MyStationerySupplies Data Warehouse

Data Warehouses are usually designed using either the **Star** or **Snowflake** Schema. You have decided to adopt the Star Schema for the design of MyStationerySupplies' data warehouse. Your first draft of the **star schema** is shown below:



Section 3: Table Specifications

The following shows the specification of each table:

Customer Dimension

Customer		
Column Name	Data Type	Allow Nulls
Customer_Key	int	<input type="checkbox"/>
Customer_Name	nvarchar(255)	<input type="checkbox"/>
Address	nvarchar(255)	<input checked="" type="checkbox"/>
Phone_Number	nvarchar(255)	<input checked="" type="checkbox"/>

Product Dimension

Product		
Column Name	Data Type	Allow Nulls
Product_Key	int	<input type="checkbox"/>
Description	nvarchar(255)	<input type="checkbox"/>
Category	nvarchar(100)	<input type="checkbox"/>
Brand	nvarchar(100)	<input type="checkbox"/>

Region Dimension

Region		
Column Name	Data Type	Allow Nulls
Region_Key	int	<input type="checkbox"/>
Region_Code	nvarchar(50)	<input type="checkbox"/>
Description	nvarchar(255)	<input type="checkbox"/>

Time Dimension

Time		
Column Name	Data Type	Allow Nulls
Time_Key	int	<input type="checkbox"/>
Year	int	<input type="checkbox"/>
Quarter	int	<input type="checkbox"/>
Month	int	<input type="checkbox"/>
Date	datetime	<input type="checkbox"/>
Day	int	<input type="checkbox"/>
DayOfWeek	nvarchar(100)	<input type="checkbox"/>

Vendor Dimension

Vendor		
Column Name	Data Type	Allow Nulls
Vendor_Key	int	<input type="checkbox"/>
Vendor_Name	nvarchar(255)	<input type="checkbox"/>
Address	nvarchar(255)	<input checked="" type="checkbox"/>
Phone_Number	nvarchar(100)	<input checked="" type="checkbox"/>

Sales Facts (Facts Table)

SalesFacts		
Column Name	Data Type	Allow Nulls
Time_Key	int	<input type="checkbox"/>
Product_Key	int	<input type="checkbox"/>
Vendor_Key	int	<input type="checkbox"/>
Customer_Key	int	<input type="checkbox"/>
Region_Key	int	<input type="checkbox"/>
Quantity	int	<input type="checkbox"/>
Price	decimal(10, 2)	<input type="checkbox"/>

Section 4: Summary of tasks

Complete the tasks as stated below

No.	Task
a)	Create a new database in MS SQL Server called "BI_Prac_1_DW"
b)	Create all tables shown in the star schema above (Refer to Appendix A if you are unsure how to create a table)
c)	Set the all foreign keys as shown (Refer to Appendix B if you are unsure how to create a foreign key)
d)	Download and execute "addData_Prac_1.sql_Part1.sql" and "addData_Prac_1_Part2.sql" to create records in the database tables.
e)	Come up with SQL statements that can help answer the following sales related questions: (i) Who are the top 3 customers and how much are their sales revenues? (ii) What is the sales revenue generated from each region? (iii) Which are the top 5 selling products in terms of revenue earned? (iv) Which is the number one brand in terms of quantity sold? (v) What is the top-grossing product for each vendor?
f)	Here is one example of how you can answer the question "Who are the top 3 customers?"
	<pre> SELECT Top (3) Customer.Customer_Name, SUM(SalesFacts.Price * SalesFacts.Quantity) AS [Sales Total] FROM Customer, SalesFacts WHERE Customer.Customer_Key = SalesFacts.Customer_Key GROUP BY [Customer_Name] ORDER BY [Sales Total] DESC; </pre>

Note:

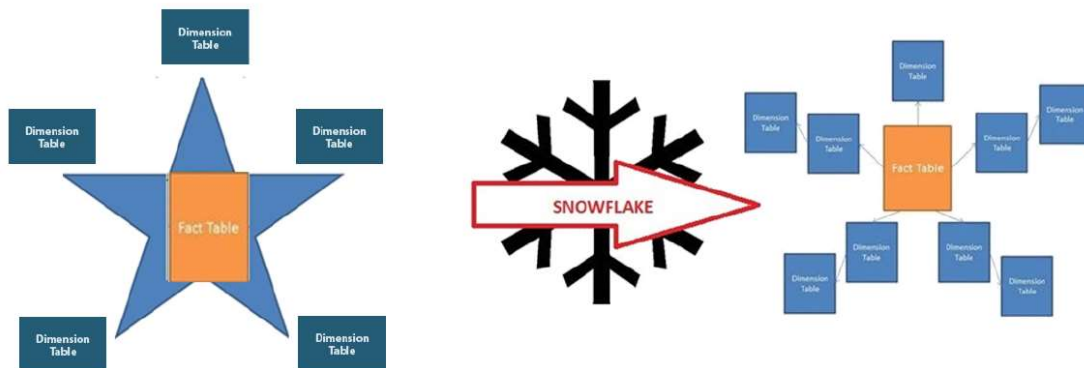
1. The 'addData_Prac_1-Part2.sql' should insert 4807 SalesFacts records

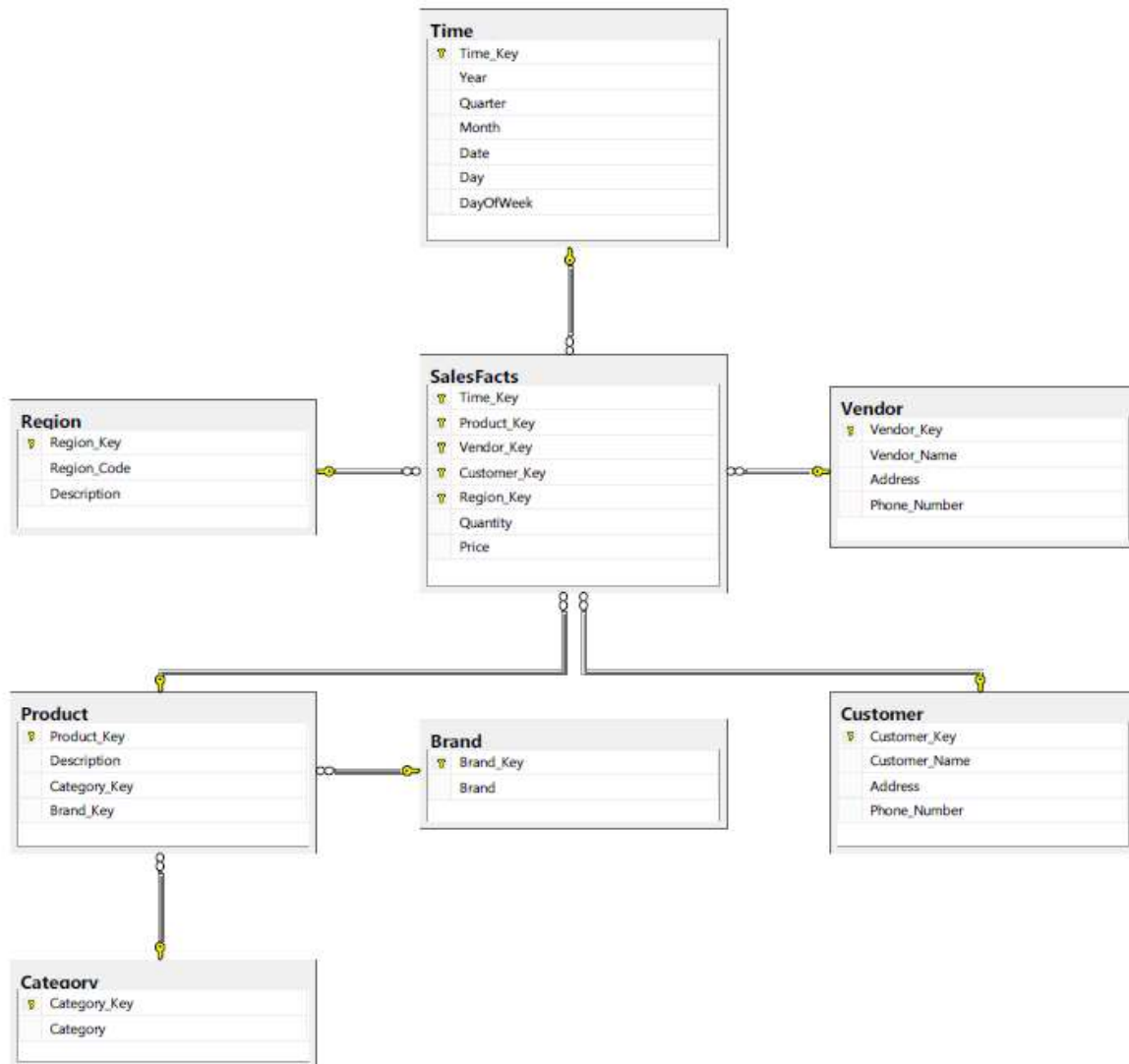
Section 5: Snowflake Schema for MyStationerySupplies Data Warehouse

You will create a similar database on Microsoft SQL Server, but this time, you will use a **Snowflake** schema instead.

The Snowflake schema is a more complex data warehouse model than a Star schema. It is called a snowflake schema because the diagram of the schema resembles a snowflake. Snowflake schemas normalize dimensions to eliminate redundancy (repeating data). That is, the dimension data is broken up into multiple tables instead of one large table.

The next page shows the Snowflake Schema of the database which you will create in this practical. Notice that the Product Dimension table is now broken into 2 smaller tables. Hence, for this practical, you will be creating 1 Fact Table, and 7 Dimension Tables instead.





Section 6: Table specifications

The following shows the specification of each table:

Customer Dimension

Customer			
	Column Name	Data Type	Allow Nulls
PK	Customer_Key	int	<input type="checkbox"/>
	Customer_Name	nvarchar(255)	<input type="checkbox"/>
	Address	nvarchar(255)	<input checked="" type="checkbox"/>
	Phone_Number	nvarchar(255)	<input checked="" type="checkbox"/>

Product Dimension

Product			
	Column Name	Data Type	Allow Nulls
PK	Product_Key	int	<input type="checkbox"/>
	Description	nvarchar(255)	<input type="checkbox"/>
	Category_Key	int	<input type="checkbox"/>
	Brand_Key	int	<input type="checkbox"/>

Brand Dimension

Brand			
	Column Name	Data Type	Allow Nulls
PK	Brand_Key	int	<input type="checkbox"/>
	Brand	nvarchar(255)	<input type="checkbox"/>

Category Dimension

Category			
	Column Name	Data Type	Allow Nulls
PK	Category_Key	int	<input type="checkbox"/>
	Category	nvarchar(255)	<input type="checkbox"/>

Region Dimension

Region			
	Column Name	Data Type	Allow Nulls
PK	Region_Key	int	<input type="checkbox"/>
	Region_Code	nvarchar(50)	<input type="checkbox"/>
	Description	nvarchar(255)	<input type="checkbox"/>

Time Dimension

Time			
	Column Name	Data Type	Allow Nulls
PK	Time_Key	int	<input type="checkbox"/>
	Year	int	<input type="checkbox"/>
	Quarter	int	<input type="checkbox"/>
	Month	int	<input type="checkbox"/>
	Date	datetime	<input type="checkbox"/>
	Day	int	<input type="checkbox"/>
	DayOfWeek	nvarchar(100)	<input type="checkbox"/>

Vendor Dimension



Vendor			
	Column Name	Data Type	Allow Nulls
PK	Vendor_Key	int	<input type="checkbox"/>
	Vendor_Name	nvarchar(255)	<input type="checkbox"/>
	Address	nvarchar(255)	<input checked="" type="checkbox"/>
	Phone_Number	nvarchar(100)	<input checked="" type="checkbox"/>

Sales Facts (Facts Table)

SalesFacts			
	Column Name	Data Type	Allow Nulls
	Time_Key	int	<input type="checkbox"/>
	Product_Key	int	<input type="checkbox"/>
	Vendor_Key	int	<input type="checkbox"/>
	Customer_Key	int	<input type="checkbox"/>
	Region_Key	int	<input type="checkbox"/>
	Quantity	int	<input type="checkbox"/>
	Price	decimal(10, 2)	<input type="checkbox"/>

Section 7: Summary of tasks

Complete the tasks as stated below

No.	Task
a)	Create a new database in MS SQL Server called "BI_Prac_2_DW".
b)	Create all tables shown in the Snowflake schema above.
c)	Set the all foreign keys as shown
d)	<p>Compare the table structure of Practical 1 (Star Schema) to that of Practical 2 (Snowflake Schema)</p> <p>You will see that the Snowflake schema has more dimension tables than the Star Schema. This is because some of the information in the primary dimension tables Product Table has been moved to the secondary dimension tables Brand and Category.</p> <div>   </div>
e)	Download and execute "addData_Prac_2_Part1.sql", "addData_Prac_2_Part2.sql" and "addData_Prac_2_Part3.sql" to create records in the database tables.
f)	<p>Come up with SQL statements that can help answer the following sales related questions:</p> <ul style="list-style-type: none"> (i) Who are the top 3 customers and how much are their sales revenues? (ii) What is the sales revenue generated from each region? (iii) Which are the top 5 selling products in terms of revenue earned? (iv) Which is the number one brand in terms of quantity sold? (v) What is the top-grossing product for each vendor?

Question:

- Is there any difference in the Query statements for task (f), written for Star and Snowflakes schema?

Appendix A: How to create a table using script

No.	Task
a)	<p>You can create tables in SQL Server either via script or the GUI table designer</p> <p>The advantage of using scripts to create the tables is that you can easily drop the tables and recreate them, if required</p>
b)	<p>Example 1 shows a script of how you can create the Customers table in this Practical as well as setting the Primary Key for it. You can use this script as a base to create the tables such as the Product Table etc.</p>
c)	<p>Note that the Primary key for the SalesFacts table is different from the rest of the tables. It is a composite key made up of the Primary Keys of the 5 dimension tables. To create a composite key made up of other primary keys, refer to Example 2 below.</p>
Example 1 : Create a Table with one primary key	
<pre>CREATE TABLE Customer (Customer_Key int NOT NULL, Customer_Name varchar(50) NOT NULL, Address varchar(99) NOT NULL, Phone_Number varchar(25) NOT NULL, PRIMARY KEY (Customer_Key));</pre>	
Example 2: Create a Table whose primary key is a composite one made up of 5 other keys residing in the same table	
<pre>CREATE TABLE SalesFacts (Time_Key int NOT NULL, Product_Key int NOT NULL, Vendor_Key int NOT NULL, Customer_Key int NOT NULL, Region_Key int NOT NULL, Quantity int NOT NULL, Price decimal(10,2) NOT NULL, CONSTRAINT pk_SalesFactKey PRIMARY KEY (Time_Key,Product_Key,Vendor_Key,Customer_Key,Region_Key));</pre>	

Appendix B: How to set foreign keys

No.	Task
a)	Below is a sample script of how you can create add the Customer_Key as a foreign key to the SalesFacts table in this Practical, after the SaleFacts table has been created
Method 1: Add in the foreign key after the table has been created	
<pre>ALTER TABLE SalesFacts ADD FOREIGN KEY (Customer_Key) REFERENCES Customer(Customer_Key);</pre>	
Method 2: Add in the foreign key at the time of creation of the table has been created	
<pre>CREATE TABLE SalesFacts (Time_Key int NOT NULL, Customer_Key int FOREIGN KEY REFERENCES Customer(Customer_Key), Product_Key int NOT NULL, Vendor_Key int NOT NULL, Region_Key int NOT NULL, Quantity int NOT NULL, Price decimal(10,2) NOT NULL, CONSTRAINT pk_SalesFactKey PRIMARY KEY (Time_Key, Product_Key, Vendor_Key, Customer_Key, Region_Key),</pre>	

Appendix C: Using the Partition By with Rank()

What Is PARTITION BY?

Partitioning is optional part of a SQL window function command. Most analytical functions, including **RANGE**, can be written with a **PARTITION BY** clause:

What does a **PARTITION** do? It defines a *partition* on which the window function does some work (e.g. executes).

It is important to note the similarities between **GROUP BY** and **PARTITION BY** parts of the SQL statements. But be careful not to confuse the two. GROUP BY defines aggregation groups on the level of the SQL statement and **PARTITION BY** defines virtual PARTITIONS that are needed for analytical function output.

Ranking Functions

Ranking analytical functions rank data based on the ordering you specify in the function specification **inside the specified rank partition**.

We can now redefine **RANK** as a window function that returns a **rank** or a number based on the ordering of the rows by some condition **inside** a predefined **partition**.

Example

In our SalesFact, we want to find out the total quantity purchased by each customer for the various kind of products. We want to **rank the largest quantity purchased to the smallest quantity purchased for each customer**.

SQL code

```
select customer_key, product_key,
       sum(quantity) as TotalPurchased,
       rank() over (partition by customer_key order by sum(quantity) desc)
as rank
from salesfacts
group by customer_key, product_key
order by customer_key, sum(quantity) desc
```

If we want to be more specific, we can write as:

```
select customer_name, description,
       sum(quantity) as TotalPurchased,
       rank() over (partition by customer_name order by sum(quantity) desc)
       as rank
from salesfacts SF
     join customer C on SF.customer_key = C.customer_key
     join Product P on SF.product_key = P.product_key
group by customer_name, description
order by customer_name, sum(quantity) desc
```

If you do not want to show all but only the top 3, you can use the following with RankedTable as a virtual table.

```
with RankedTable
(CustomerID, ProductID, TotalPurchased, myRank) as
  (select customer_key, product_key,
        sum(quantity) as TotalPurchased,
        rank() over
          (partition by customer_key order by sum(quantity) desc) myRank
   from salesfacts
   group by customer_key, product_key)

select CustomerID, ProductID, TotalPurchased, myRank
   from RankedTable
  where myRank in (1,2,3)
```

Alternatively, write a longer query using SubQuery.

```
select customer_key, product_key, totalPurchased, myRank
   from
     (Select customer_key, product_key, sum(quantity) as totalPurchased,
           rank() over
             (partition by customer_key order by sum(quantity) desc) myRank
      from salesfacts
      group by customer_key, product_key) as RankedTable
 where myRank in (1,2,3)
```