He Tong 867488

Wang Yao 869992

January 16, 2019

ISYS90086 Assignment 1 – Data Warehouse Design Assignment

Fantastic Fireworks Case Study

Table of Contents

1. Executive Summary

Data used to be just data, but now we call ‘big data’, ‘strategic data’, ‘multi-structured data’. Likewise, the chief goals of all the companies today is to enable data-driven decisions and actions(Mallach,2000). Data warehousing (DW) is considered to be tailored to help upper management make these decisions intelligently and reliably by providing aggregated, multi-dimensional, user-oriented, strategic information and shielding complex of query coding for end-users.

Fantastic Fireworks, as a highly profitable business in recent years, is perfectly qualified for and has the necessity of evolving into data warehouse. Look at these business expansion ambitions: opening branches, expanding the floor space, optimizing store opening times and adjusting the exhibition of best-selling products, none of these cannot be solved by a good data warehouse model. As such, a star schema data warehouse model is designed for the company based on the current needs and two operational systems. This report presents a discussion of the data warehouse concept, necessity, and benefits it can bring to organizations, followed by design details along with a corresponding diagram of the data warehouse, and finally demonstrates the way in which DW may be used to address some business problems. Data Dictionary, SQL statement and work Breakdown are included in the appendix for a further detailed view of the design.

2. Introduction to Data Warehousing

2.1. Concept of Data Warehousing:

Over the decades, companies increasingly focus on how to integrate large and heterogeneous data in their business intelligence models so that they can predict sales figures and make strategic decisions. Data warehouse is considered tailored to these needs. It is a blend of technologies and components for aggregating as well as

processing current and historical data from varied sources without hindering operational systems (Inmon, Welch & Glassey,1997), to present greater executive insight into corporate performance, thus rendering the strategic decision making easy and intelligent.

2.2. Necessity of Data Warehousing

For the business expansion and long-term development of the company, the basis of business decision-making should no longer be empiricism but analysis of big data(Todman,2001). The management team of the company including the group accountant, inventory manager currently do knowledge tasks such as bestseller analysis or reordering changes relying on their intuition or experience instead of aggregated data, which increases the risk of making wrong decisions as the company scales up.

2.3. Benefits of Data Warehousing

Data warehouses are effectively read-only databases compared to legacy systems. Traditional database design techniques such as ER modeling and normalization aim at inputting and updating data efficiently and reliably into databases (Jarke, Lenzerini, Vassiliou & Vassiliadis, 2003). However, on the downside, these methods are too daunting for end users (probably not SQL specialists) to understand and write queries against when complex statistical analyses are involved. Besides, multi-table joins involved considerably reduce query efficiency. Data warehouses just address these limitations; they are, end-user tailored and complex retrieval oriented. Given that the management staff in your company have rich business experience but not IT expertise, this user-friendly system does suit it.

Data warehouse gives a holistic, integrated view of company data. Information from current systems is large and heterogeneous; They are difficult to integrate and aggregate typically as they have different data formats and have inconsistent problems

getting from multiple independent locations. Data warehouses introduce data processing technologies (ETL etc.) to precisely output archives, derived, summarized information for heuristic use. It renders the corporate information consistent and guarantees data quality, making strategic decisions much more reliable and convincing accordingly.

The pivotal development, dimensional modeling, provides multi-dimensions for different perspective of business intelligent analysis. The insights about product, customers, store location and opening times now needed for decision-making by the senior manager is exactly what data warehouse system is devoted to. It helps a lot ranging from supply chain adjustment, customer selection, human capital, financial performance, and pricing.

3. Design of the Data Warehouse

As the Figure 1 shows below, a simple star schema is created for the data warehouse with sales fact table in the center and five dimensions tables including Customer, Employee, Store, Product and Time along the spikes of the star, which intuitively answers the questions of what, when, where, by whom, and to whom. Each dimension table is related to the fact table in a one-to-many relationship. The design details are as follows.

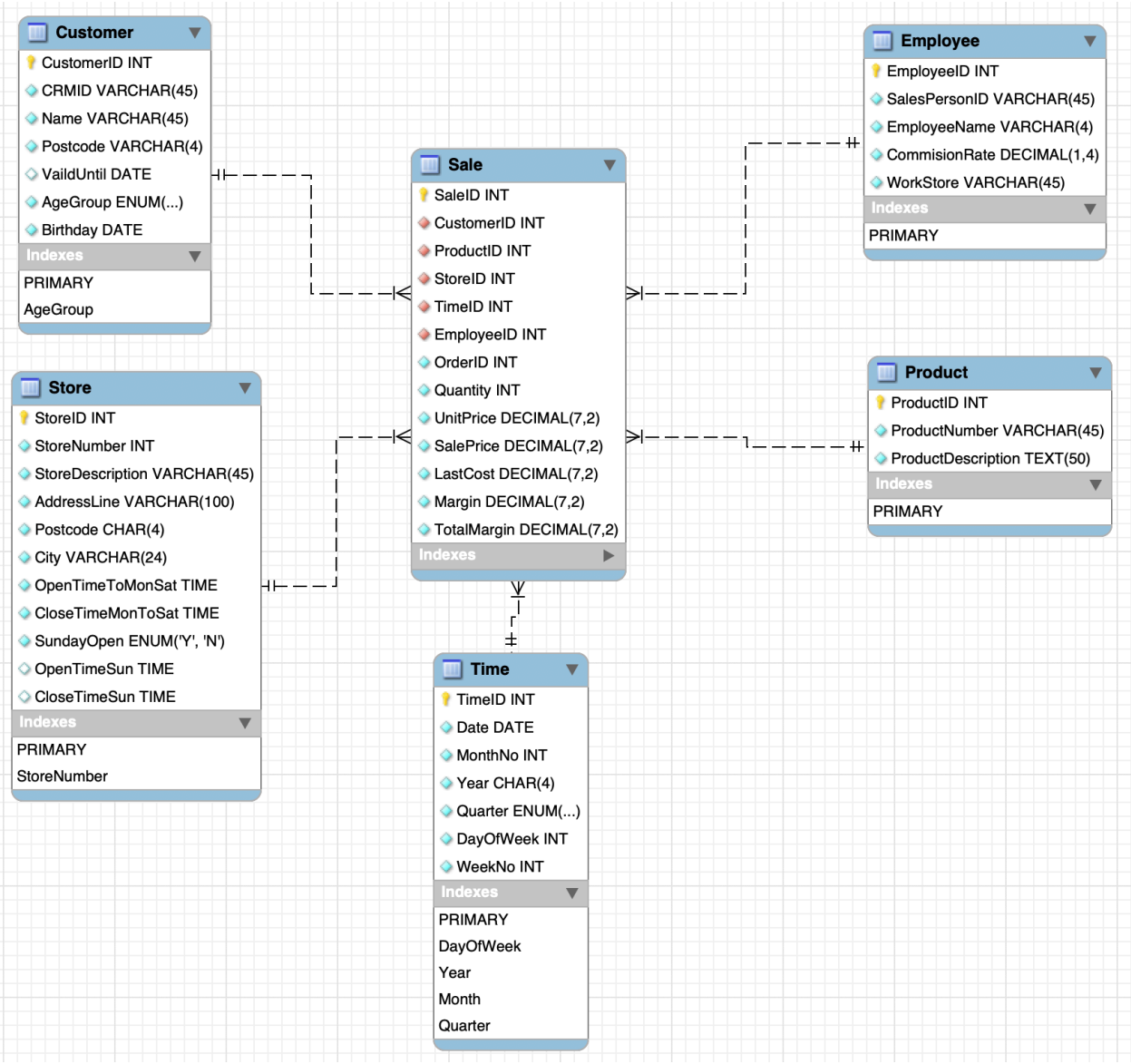

Figure 1

3.1. Fact table

Only one fact table, the sales table is needed. Source data comes from two different operational systems: the inventory system and the sales system.

In light of the needs of insights about products, customers, store location, and opening times and five profit-related questions concerning with five aspects (customers, products, store location, periods and employees), we form a sales fact table to link them all so that upper management can easily visualize the answers. Note that the cost of the product on the most recent order before the sale (LastCost) from inventory system is directly as an attribute in the sales fact table after ETL processing. Other data in inventory systems are not related to what management care about. Specifically, the fact table has a surrogate key SaleID to sequence each row of a transaction; accordingly, OrderID which is the sales system primary key is kept as a non-key attribute in the fact table to avoid built-in meanings. Five foreign keys including CustomerID, ProductID, StoreID, TimeID and EmployeeID are the respective primary keys in the dimension tables. The rest are semi-addictive attributes, including Quantity (The quantity of one kind of product sold in a order), Unitprice (adviced price), SalePrice (selling price), LastCost derived from operational systems, Margin (computed by SalePrice minus LastCost) and TotalMargin (product of Quantity and Margin). TotalMargin is fully addictive, which will be used frequently in the later questions.

The facts and dimensions are at the natural lowest grain, that is, a single product, an individual customer and a certain employee, a specific certain date, and a nominated store. In this arrangement, first, it satisfies the details of the five questions; second, users can drill down and roll up in many different ways without the need to go to the source systems and add new attributes or new dimensions without occurrence of unequal data error.

3.2. Product table

Product table has the least number of attributes among the dimensions, including ProductID, ProductNumber, ProductDescription, and ValidFrom (no product name in the source system). ProductID is a surrogate key, and the rest correspond to PartNumber, Description, Price and Validfrom in the inventory system.

Unit price of product is removed to the product table but exists in the fact table. This is because adding price attribute in the product table will increase the burden on the system as price would change after its valid date, and if that, a new row will be added with all original, but a new price and a new product id and historical records still remain in case of future needs.

Any attribute about order is not in this dimension, for example, Order date, order quantity, supplier, etc. This is because, first, the product table would become big both the more columns and several rows added after a cost change; second, the managers have to join a dimension with the fact table and always with the product table to get cost every time they want to know profit things; third, five questions the manager cares about do not refer to any order information except the most recent cost as mentioned before.

It is worth noting that, in practice, Product and Sales are in a many-to-many relationship, that is, one transaction may involve several types of products, and one type of product is related to massive transactions. The arrangement here is to split each transaction into rows each of which only has one type of an item, thus making Product table related to Sales fact table in a one-to-many relationship.

3.3. Customer table

Customer table is to keep the customer information, consisting of seven attributes, CustomerID, CRMID, Name, Postcode, ValidUntil, AgeGroup, and Birthday. CRMID is the primary key in the CUSTOMER table of the source system, which has built-in meanings: for example, ‘C330Bris’ of which the last four digit represents the customer’s nominated store. Another possibility is that one customer numbers in the production system may be reassigned to a customer if the previous owner is no longer listed with the company. Besides, Postcode is a slowly changing attribute as customer may change their home address over time. Here we keep the old values by creating a new row for each change. For avoiding build-in meanings, customer numbers conflict, and history handling of slowly changing dimensions, CustomerID is created as the surrogate key in the Customer dimension table.

ValidUntil records the date time on which one customer changes his address, a row with the new address to be added with that the old row of the old address still exists. This attribute helps determine the suburb in which customers lived at the time of any particular sale.

Age Group is a not null attribute aggregated customers into three groups-30 years and under, 31-50, and over 50. The manager would know the preferences of people of different ages. Birthday of the customer is not mandatory to the table but, if the manager proposes a new age criterion, that would be useful.

3.4. Store table

Store table keeps three aspects of store information: identification (StoreID, StoreNumber, StoreDescription), location (AddessLine, Postcode, City) and opening times (OpenTimeMonToSat, CloseTimeMonToSat, SundayOpen, OpenTimeSun, CloseTimeSun) with the surrogate key StoreID as the manager is thinking about adjustments of Opening times in the future. OpenTimeMonToSat and CloseTimeMonToSat are open and close time point Monday through Saturday, and OpenTimeSun and CloseTimeSun are on Sunday, in the datatype of TIME (HH:MM: SS). SundayOpen is an enum datatype, ‘Yes’ or "No’ to be chosen to represent if the store opens on Sunday. Since only the original store opens on Sunday, the values in OpenTimeSun and CloseTimeSun along with the other three store records are null. In this design, the manager can analyze and optimize store opening times, for example, opening on Sundays at other locations by easily visualizing the profits made on Monday to Sunday or on Sunday.

3.5. Time table

Time table has 7 attributes, TimeID, Date, MonthNo, Year, Quarter, DayOfWeek and WeekNo, TimeID as the primary key. MonthNo fills in one integer among 1-12; Year is a string of length 4 like ‘2018’; Quarter chooses ‘1’, ‘2’, ‘3’, ‘4’ to show which quarter is in; DayOfWeek uses one integer among 1-7 to match Mon-Sun; WeekNo stores which week the time is in. The manager is easy to analyse profits aggregated by various time units.

3.6. Employee table

This table contains EmployeeID, SalespersonID, EmployeeName, WorkStore and CommissionRate, EmployeeID (simply system-generated sequence numbers) as the surrogate key. SalespersonID matches sales person ID from the source system. Particularly, the attribute WorkStore records which store a salesperson works at, which is extracted from SalespersonID (the first letter of SalespersonID 'B' 'D' 'M' 'S' corresponding to 'Brisbane', 'Darwin', 'Melbourne', 'Sydney'). CommissionRate is a percentage of the sales price of an item, which means a salesperson sell a piece of product A and will get dollar of his/her commission rate multiplying sales price of A.

This table helps find the sales of each salesperson with utmost ease, say in a period of time, how much money each salesperson sells? What is the number of a certain product each salesperson sells? How much commission each will earn etc.

4. Data warehousing implementation in Business Problems

4.1. Who are the key customers?

Data retrieved:

Data to be retrieved among 'CustomerID, Quantity, UnitPrice, SalePrice, LastCost, Margin, TotalMargin' from Sale table, 'CustomerID, CRMID, Postcode, AgeGroup, ValidUntil' from Curstomer table and 'TimeID, Year, MonthNo, Quarter'.

Key points:

- Join Sale table to Customer table and Time table via the respective primary key 'TimeID' and 'CustomerID'.

- Choose a period of time according to user need, such as picking up all the rows where ‘Year’ = 2018, or ‘Quarter’ = 3, etc.
- Sum the amount of ‘Quantity’ with the same ‘CustomerID’ to get the total number each customer purchased.
- For each customerID with every item, compute the sum of product of its corresponding ‘SalesPrice’ and ‘Quantity’ (has been replaced as the quantity of a single item a customer buys), as a new attribute ‘OneCusSpent’.
- The most important thing to note is that different ‘CustormerID’ may point to the same customer as a new row with a new ‘CustormerID’ will be created each time the customer changes his/her address. Assuming the retrieved row is formed as ‘CustomerID, CRMID, OneCusSpent, Quarter, Postcode’, and there are two rows ‘100, C4233Bris, 555.55, 3, 4039’ and ‘700, C4233Bris, 3, 4006’, among which, ‘100’ and ‘700’ belong to the same person. Thus, whether to group by ‘CRMID’ to merge rows of the same person with different address depends on the specific requirements of need.

The way to support decision-making:

There are various criteria to define what a key customer is, say who buys the most frequently, who buys the largest number of the products, who spends the most money or who brings the highest margin to the company in a certain quarter, month or year. If users do not care about where customers are living when they purchase, group by ‘CRMID’ at last to merge all rows of a single customer and:

- descending by ‘OneCusSpent’, users will see the top N customers who spent the most money in a given period of time and how much they spent at a glance;
- descending by ‘TotalMargin’, users will see the top N customers who bring the highest margin to the company in a period;
- From other perspective like who buys the largest number of the products, just descend by ‘Quantity’;

- If the user wants to see data in various period of time, just change the condition statement in the time-filtering part.

Else, if users want to know the changes in purchasing after one customer moves and want to see the correponding address, they can just analyze based on ‘CustomerID’.

4.2. Which products are the most profitable?

Data retrieved:

Data that needs to be retrieved among ‘Productid, Quantity, UnitPrice, SalePrice, LastCost, Margin, TotalMargin’ from Sale table, ‘Product ID, ProductNumber, ProductDescription’ from Product table and ‘TimeID, Year, MonthNo, Quarter’ from Time table.

Key points:

- In the design, we use ETL process to intelligently match the most recent cost of each item from operational system. Join Sale table with Time table by ‘TimeID’ and Product table by ‘ProductID’.
- Choose one of the attributes among ‘Year’, ‘Quarter’, ‘MonthNo’ as required.
- Sum ‘TotalMargin’ in each ‘ProductID’ named ‘ProductMargin’ and descending the rows by ‘ProductMargin’.

The way to support decision-making:

The most profitable product is the one earns most margin in a given period of time as needed. Users can easily get top N most profitable product from the results (descending by ‘ProductMargin’). They can also select ‘ProductNumber’ and ‘ProductDescription’ attributes to get further details about specific products.

4.3. Which store location is the most profitable?

Data retrieved:

In this question, the data which needs to be retrieved include ‘StoreID, Quantity, UnitPrice, SalePrice, LastCost, Margin’ from Sale table, ‘StoreID, StoreNumber, StoreDescription, AddressLine’ from Store table and ‘TimeID, Year, MonthNo’ from Time table.

Key points:

- In the design, we use ETL process to intelligently match the most recent cost of the items from operational system. Join Sale table with Time table by ‘TimeID’ and Store table by ‘StoreID’.
- Combine the two attributes ‘Year’ and ‘MonthNo’ so that data can be grouped for each month in previous year as required.
- Sum ‘TotalMargin’ (=‘Margin’ * ‘Quantity ‘) in each ‘StoreID’ named ‘StoreMargin’ and descending the rows by ‘StoreMargin’.

The way to support decision-making:

Question 3 is similar to question 2 except that the Product dimension is changed to the Store dimension. After obtaining the sorted ‘StoreID’, detailed information is showed in corresponding ‘AddressLine’, ‘Postcode’ and ‘City’ attributes in the Store table, which are the splits of the ‘Address’ attribute in the operating system.

4.4. Which time periods are the most profitable?

Data retrieved:

In this issue, the data which needs to be retrieved include ‘ProductID, Quantity, UnitPrice, SalePrice, LastCost, Margin,TotalMargin’ from Sale table and all attributes in Time table.

Key points:

- Join Sale table to Time table via the primary key ‘TimeID’.
- Select or combine attributes among ‘Year’, ‘Quarter’, ‘MonthNo’ and ‘DayOfWeek’ as required.

The way to support decision-making:

To simplify the operation of the user, the date record in the operational system has been split into multiple granularities and converted into corresponding attributes. Different attributes correspond to different time granularities, such as ‘Year’ corresponding to annual data, and ‘MonthNo’ corresponding to monthly data. Note that, there is a ‘DayOfWeek’ attribute in the Time table to indicate the day of the week. Values 1 to 7 correspond to Sunday to Saturday, respectively. Therefore, by adding the condition ‘DayOfWeek’ == 1, data warehouse retrieves the transaction data on Sunday.

4.5. Who are the Key employees?

Data retrieved:

In this issue, the data which needs to be retrieved include ‘ProductID, Quantity, UnitPrice, SalePrice, LastCost, Margin,TotalMargin’ from Sale table, ‘EmployeeID, SalesPersonID, EmployeeName, CommissionRate, WorkStore’ from Employee table and ‘TimeID, Year, MonthNo, Quarter’.

Key points:

- Join Sale table to Employee table and Time table via the respective primary key ‘EmployeeID’ and ‘TimeID’.
- Sum the amount of ‘Quantity’ and ‘TotalMargin’ separately with the same ‘EmployeeID’ to get the total product amount and dollar each employee sold.
- For each EmployeerID with every item, compute sum of product of its corresponding ‘SalesPrice’ and ‘Quantity’ as a new attribute ‘TotalSale’.
- get the salesperson’s commission rate by selecting ‘CommissionRate’ corresponding to ‘EmployeeID’ from Employee table.

The way to support decision-making:

There are three ways to identify good employees, the number of products sold, the total price and profit. They correspond to ‘Quantity’, ‘TotalSale’ and ‘TotalMargin’. Users can

sort them on demand for different time periods. In addition, there is a ‘WorkStore’ attribute indicating which store the salesperson belongs to is extracted from ‘EmployeeID’. It helps users analyze data by employees in different stores.

References

[1] Inmon, W. H., Welch, J. D., & Glassey, K. L. (1997). *Managing the data warehouse*. John Wiley & Sons, Inc..

[2] Jarke, M., Lenzerini, M., Vassiliou, Y., & Vassiliadis, P. (2003). *Fundamentals of Data Warehouses. [electronic resource]*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. Retrieved from https://ezp.lib.unimelb.edu.au/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat00006a&AN=melb.b5446509&site=eds-live&scope=site

[3] Mallach, E. (2000). Decision support and data warehouse systems. Irwin/McGraw-Hill.

[4] Todman, C. (2001). *Designing a data warehouse : supporting customer relationship management*. Upper Saddle River, N.J. : Prentice Hall PTR, c2001. Retrieved from https://ezp.lib.unimelb.edu.au/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat00006a&AN=melb.b2844798&site=eds-live&scope=site

5. Appendix 1 – Data Dictionary

5.1. Sale fact table:

Attribute	Description	Source	Transformation
SaleID	Surrogate key unique identifier for each entry	Generated in DW	
OrderID	Unique number assigned to each purchase transaction	OrderID attribute in Sales Item table	
Quantity	The number of certain products sold in certain order	Quantity attribute in SALES ITEM table	
UnitPrice	The advertised price of product	UnitPrice attribute in PRODUCT PRICE LIST table	Choose the most recent price before the sale based on Valid from date attribute
Saleprice	The amount of money customer actually paid for unit product	SalePrice attribute in SALES ITEM table	
LastCost	The cost of this product	Cost per item attribute in PRODUCT ORDER table	using Date attribute to choose the most recent price before the order

Margin	The profit per product		= Saleprice - lastcost
TotalMargin	The profit of this kind product in this order		= Margin * Quantity

5.2. Customer table:

Attribute	Description	Source	Transformation
CustomerID	Surrogate key unique identifier for each entry	Generated in DW	
CRMID	Unique number assigned to each customer	CustomerID attribute in CUSTORMER table	
Name	The name of customers	Name attribute in CUSTOMER table	
AgeGroup	aggregate customer reports into age groups – 30 years and under, 31-50, and over 50		transform from birthday attribute of customers table
Birthday	Birthday of customer	Birthday attribute in CUSTOMER table	

Postcode	The postcode of where customer live	Postcode attribute in CUSTOMER table	
ValidUntil	When customer's info is updated	Valid until date attribute in CUSTOMER table	

5.3. Store table:

Attribute	Description	Source	Transformation
StoreID	Surrogate key unique identifier for each entry	Generated in DW	
StoreNumber	Unique number assigned to each customer	CustomerID attribute in CUSTORMER table	
StoreDescription	The description of store	Description attribute in CUSTOMER table	
AddressLine	Address of the store		transform from Address attribute of STORE table
Postcode			
City			
OpentimeMonSat	What time store opens from Monday to Saterday	Case study document	
ClosetimeMonSat	What time store closes from	Case study document	

	Monday to Saterday		
SundayOpen	whether store opens on Sunday	Case study document	
CloseTimeSun	What time store opens on Sunday	Case study document	
OpenTimeSun	What time store closes on Sunday	Case study document	

5.4. Time table:

Attribute	Description	Source	Transformation
TimeID	Surrogate key unique identifier for each entry	Generated in DW	
Date	The time order happened	Date attribute in SALES table	
MonthNo	The date in different grain		Split from Date
Year			
Quarter			
DayOfWeek			
WeekNo			

5.5. Employee table:

Attribute	Description	Source	Transformation
EmployeeID	Surrogate key unique identifier for each entry	Generated in DW	

SalepersonID	Unique number assigned to each sales person	ID attribute in SALES PERSON table	
EmployeeName	Name of the sales person	Name attribute in SALES PERSON table	
Commissionerate	Percentage of commission received per item sold	Commission Rate attribute in SALES PERSON table	
WorkStore	Which store the sale person belongs to		The first character of SalepersonID

5.6. Product table：

Attribute	Description	Source	Transformation
ProductID	Surrogate key unique identifier for each entry	Generated in DW	
ProductNumber	Unique number assigned to each product	ID attribute in PRODUCT table	
ProductDescription	Description about products	Description attribute in PRODUCT table	

6. Appendix2 – SQL

6.1. SQL Statement:

```
-- MySQL Script generated by MySQL Workbench
-- Wed Jan 16 01:25:01 2019
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Store`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Store` (
  `StoreID` INT NOT NULL,
  `StoreNumber` INT NOT NULL,
  `StoreDescription` VARCHAR(45) NOT NULL,
  `AddressLine` VARCHAR(100) NOT NULL,
  `Postcode` CHAR(4) NOT NULL,
  `City` VARCHAR(24) NOT NULL,
  `OpenTimeToMonSat` TIME NOT NULL,
  `CloseTimeMonToSat` TIME NOT NULL,
  `SundayOpen` ENUM('Y', 'N') NOT NULL,
  `OpenTimeSun` TIME NULL,
  `CloseTimeSun` TIME NULL,
  PRIMARY KEY (`StoreID`),
  INDEX `StoreNumber` (`StoreNumber` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (
  `CustomerID` INT NOT NULL,
  `CRMID` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Postcode` VARCHAR(4) NOT NULL,
  `VaildUntil` DATE NULL,
```

```
  `AgeGroup` ENUM('-30', '31-50', '50-') NOT NULL,
  `Birthday` DATE NOT NULL,
  PRIMARY KEY (`CustomerID`),
  INDEX `AgeGroup` (`AgeGroup` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Time`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Time` (
  `TimeID` INT NOT NULL,
  `Date` DATE NOT NULL,
  `MonthNo` INT NOT NULL,
  `Year` CHAR(4) NOT NULL,
  `Quarter` ENUM('1', '2', '3', '4') NOT NULL,
  `DayOfWeek` INT NOT NULL,
  `WeekNo` INT NOT NULL,
  PRIMARY KEY (`TimeID`),
  INDEX `DayOfWeek` (`DayOfWeek` ASC),
  INDEX `Year` (`Year` ASC),
  INDEX `Month` (`MonthNo` ASC),
  INDEX `Quarter` (`Quarter` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Product` (
  `ProductID` INT NOT NULL,
  `ProductNumber` VARCHAR(45) NOT NULL,
  `ProductDescription` TEXT(50) NOT NULL,
  PRIMARY KEY (`ProductID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Employee` (
  `EmployeeID` INT NOT NULL,
  `SalesPersonID` VARCHAR(45) NOT NULL,
  `EmployeeName` VARCHAR(4) NOT NULL,
  `CommisionRate` DECIMAL(1,4) NOT NULL,
  `WorkStore` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`EmployeeID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Sale`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Sale` (
```

```
  `SaleID` INT NOT NULL,
  `CustomerID` INT NOT NULL,
  `ProductID` INT NOT NULL,
  `StoreID` INT NOT NULL,
  `TimeID` INT NOT NULL,
  `EmployeeID` INT NOT NULL,
  `OrderID` INT NOT NULL,
  `Quantity` INT NOT NULL,
  `UnitPrice` DECIMAL(7,2) NOT NULL,
  `SalePrice` DECIMAL(7,2) NOT NULL,
  `LastCost` DECIMAL(7,2) NOT NULL,
  `Margin` DECIMAL(7,2) NOT NULL,
  `TotalMargin` DECIMAL(7,2) NOT NULL,
  PRIMARY KEY (`SaleID`),
  INDEX `fk_Sale_Customer_idx` (`CustomerID` ASC),
  INDEX `fk_Sale_Product1_idx` (`ProductID` ASC),
  INDEX `fk_Sale_Store1_idx` (`StoreID` ASC),
  INDEX `fk_Sale_Time1_idx` (`TimeID` ASC),
  INDEX `fk_Sale_Employee1_idx` (`EmployeeID` ASC),
  CONSTRAINT `fk_Sale_Customer`
    FOREIGN KEY (`CustomerID`)
    REFERENCES `mydb`.`Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_Product1`
    FOREIGN KEY (`ProductID`)
    REFERENCES `mydb`.`Product` (`ProductID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_Store1`
    FOREIGN KEY (`StoreID`)
    REFERENCES `mydb`.`Store` (`StoreID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_Time1`
    FOREIGN KEY (`TimeID`)
    REFERENCES `mydb`.`Time` (`TimeID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_Employee1`
    FOREIGN KEY (`EmployeeID`)
    REFERENCES `mydb`.`Employee` (`EmployeeID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`category`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`category` (
  `category_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`category_id`));
```

```
-- -----------------------------------------------------
-- Table `mydb`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`user` (
  `username` VARCHAR(16) NOT NULL,
  `email` VARCHAR(255) NULL,
  `password` VARCHAR(32) NOT NULL,
  `create_time` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP);


-- -----------------------------------------------------
-- Table `mydb`.`timestamps`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`timestamps` (
  `create_time` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `update_time` TIMESTAMP NULL);


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

6.2. Index explanation:

‘Year’, ‘MonthNo’ and ‘Quarter’ are set to index respectively. Because the time dimension applies to most situations, such as users accessing different data by year, quarter, or month. Designing these properties as directories can speed up queries.

As discussed above, the ‘CRMID’ in the customer table may be queried instead of the surrogate key, so they can be set to index.

The Customer table is relatively large, and the ‘AgeGroup’ attribute has only three values. When a user wants to divide user data by age group, it is useful to set it as a index.

7. Appendix 3 – Work Breakdown

The work breakdown structure is shown in Figure 2. The yellow parts are what Tong He did, and the Blue parts are what Yao Wang did. The red parts are both did.

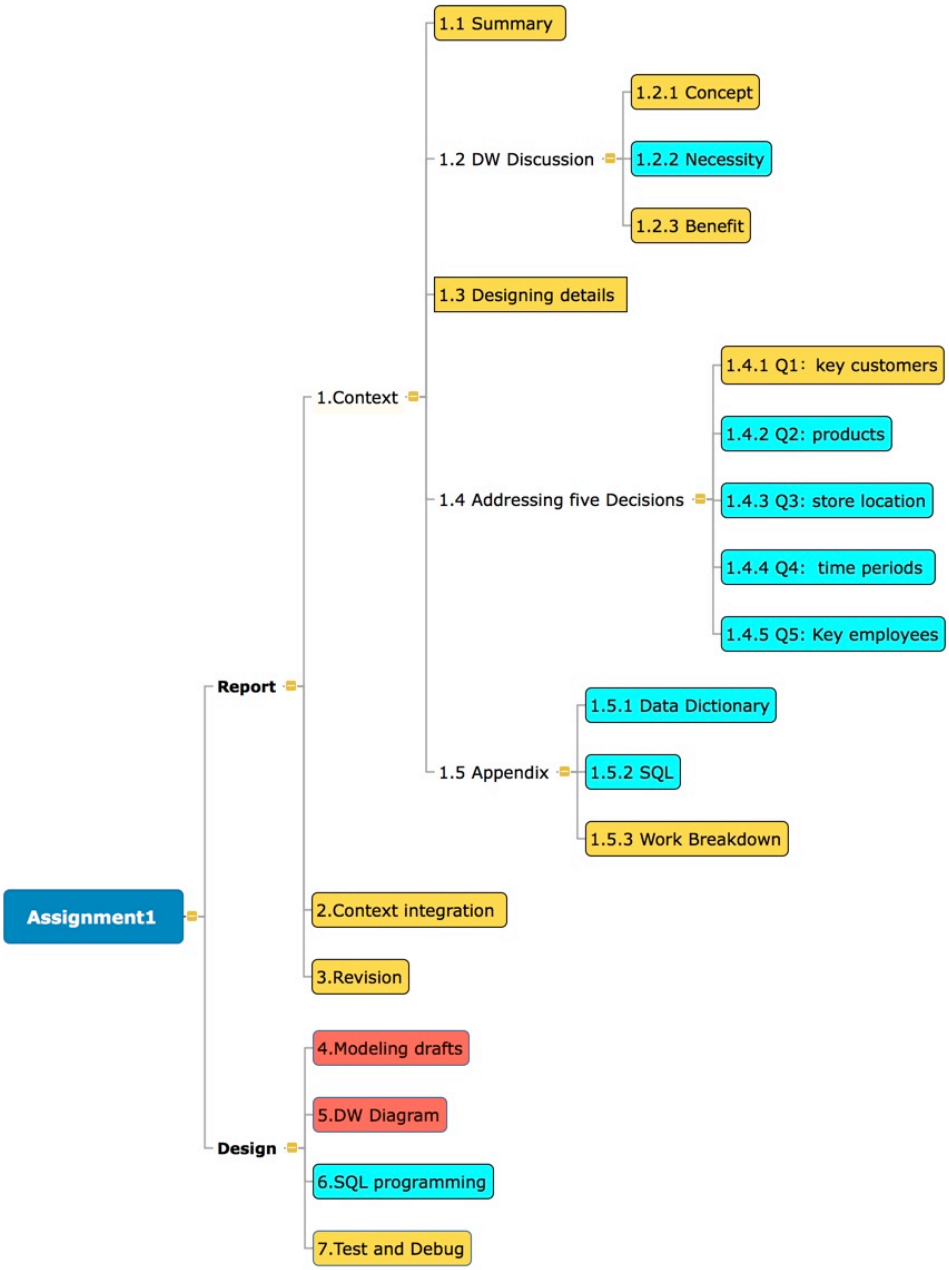

Figure 2

Name	Student id	Accomplishment
He Tong	867488	Yellow parts: 1.1 Summary 1.2.1 Concept of DW 1.2.3 Benefit of DW 1.3 Designing details 1.4.1 Q1: key customers 1.5.3 Work Breakdown 2.Context integration 3.Revision 4.Modeling drafts 5.DW Diagram 7.Test and Debug
Wang Yao	869992	Blue parts: 1.2.2 Necessity 1.4.2 Q2: products 1.4.3 Q3: store location 1.4.4 Q4: time periods 1.4.5 Q5: Key employees 1.5.1 Data Dictionary 1.5.2 SQL 4.Modeling drafts 5.DW Diagram 6.SQL programming

The serial number in the table are from *Figure 2.*