DATA WAREHOUSING

PROJECT REPORT

Group 12

Carolina Bellani M20170098, Daniela Giubilato M20170025
Piero Maggi M20170252, Alexander Obenauff M20170724

Index

Introduction to the WorldWideIMS 3

Description of Business Scenario 3

Description of Business Problem and Details of Business Needs 5

Analysing the database 5

Description of Source Systems and of Source Data 6

Description of Staging Area and of the Data Warehouse 7

Designing the Staging Area 8

- **Fact Table: "Fact_Sales"** 8
- **Dimensions** 10

Designing the Data Warehouse 15

Justification of the star schema 15

Staging Area Schema 16

Data Warehouse Schema 17

ETL Processes 18

- **Staging Area** 18
- **Data Warehouse** 20
- **Logging** 22
- **Parameters** 23

Testing 23

Validation 24

Business Intelligence recommendation 24

Conclusion 25

Introduction to the WorldWideIMS

WorldwideIMS (WWI) is a wholesale novelty goods importer and distributor operating from the San Francisco bay area.

As a wholesaler, WWI's customers are mostly companies who resell to individuals. WWI sells to retail customers across the United States including speciality stores, supermarkets, computing stores, tourist attraction shops, and some individuals. WWI also sells to other wholesalers via a network of agents who promote the products on WWI's behalf. While all of WWI's customers are currently based in the United States, the company is intending to push for expansion into other countries.

WWI buys goods from suppliers including novelty and toy manufacturers, and other novelty wholesalers. They stock the goods in their WWI warehouse and reorder from suppliers as needed to fulfil customer orders. They also purchase large volumes of packaging materials and sell these in smaller quantities as a convenience for the customers.

Recently WWI started to sell a variety of edible novelties such as chilli chocolates. The company previously did not have to handle chilled items. Now, to meet food handling requirements, they must monitor the temperature in their chiller room and any of their trucks that have chiller sections.

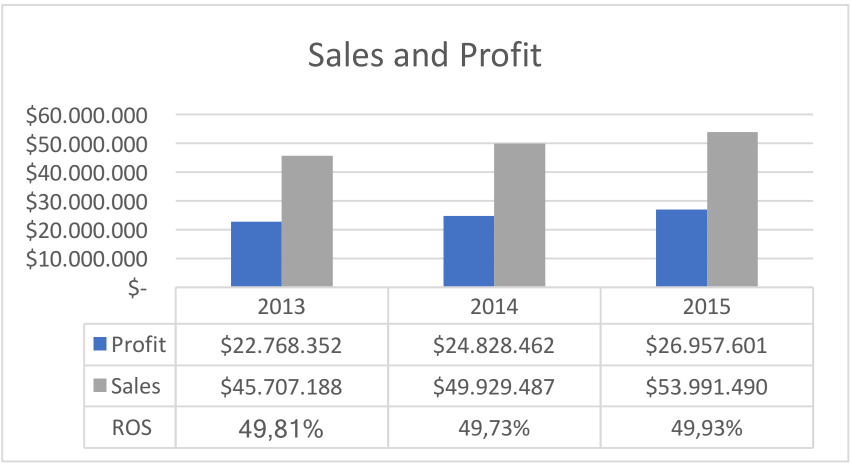
Description of Business Scenario

By studying the information about the business, we found out the following features: the company has 13 suppliers, 663 customers, and there are 19 employees, 10 of which are sales employees. The profit and sales divided for years are shown in the following table (*Table 1, Figure 1*):

Year	Profit	Sales	ROS
2013	$ 22,768,352	$ 45,707,188	49.81%
2014	$ 24,828,462	$ 49,929,487	49.73%
2015	$ 26,957,601	$ 53,991,490	49.93%
2016**	$ 11,174,766	$ 22,633,176	49.37%

** until 2016-05-31

Table 1. Profit, Sales and ROS per year.



	2013	2014	2015
■ Profit	$22.768.352	$24.828.462	$26.957.601
■ Sales	$45.707.188	$49.929.487	$53.991.490
ROS	49,81%	49,73%	49,93%

Figure 1. Profit, Sales, ROS per year

In the tables below are shown figures about the top five customers (*Table 2*) and the top five products (*Table 3*) sold per year; we can see in both the tables that there is a substantial drop in the column "SalesAmount" for the year 2016, but the reason is that we have only data until the 31-05-2016.

2013		**2014**	
CustomerName	**$SalesAmount**	**CustomerName**	**$SalesAmount**
Valter Viiding	158,195.25	Wingtip Toys [San Jacinto, CA]	155,777.10
Tailspin Toys [Indios, PR]	152,141.65	Wingtip Toys [Cuyamungue, NM]	144,079.45
Wingtip Toys [Lake Ronkonkoma, NY]	138,624.40	Cong Hoa	143,083.70
Wingtip Toys [Sarversville, PA]	137,333.65	Tailspin Toys [Netcong, NJ]	139,319.35
Bishwa Chatterjee	133,319.35	Tailspin Toys [Inguadona, MN]	137,635.60
2015		**2016**	
CustomerName	**$SalesAmount**	**CustomerName**	**$SalesAmount**
Knut Svensson	150,701.50	Tailspin Toys [Arietta, NY]	91,923.70
Laszlo Gardenier	150,506.70	Tailspin Toys [Good Hart, MI]	91,799.40
Irma Berzina	148,305.00	Wingtip Toys [Obetz, OH]	87,221.90
Wingtip Toys [Montoya, NM]	147,212.95	Wingtip Toys [North Beach Haven, NJ]	85,923.70
Svetlana Todorovic	143,382.10	Wingtip Toys [Leathersville, GA]	79,706.30

Table 2. Top 5 customers per year.

2013		**2014**	
StockItemID	**$SalesAmount**	**StockItemID**	**$SalesAmount**
Stk.Air cushion machine (Blue)	3,140,946.00	Stk.Air cushion machine (Blue)	3,207,411.00
Stk.32 mm Anti static bubble wrap (Blue)	1,719,900.00	Stk.32 mm Double sided bubble wrap 50	1,925,280.00
Stk.32 mm Double sided bubble wrap 50	1,570,240.00	Stk.32 mm Anti static bubble wrap (Blue)	1,884,750.00
Stk.10 mm Anti static bubble wrap (Blue)	1,565,190.00	Stk.10 mm Double sided bubble wrap 50	1,878,450.00
Stk.20 mm Double sided bubble wrap 50	1,545,480.00	Stk.10 mm Anti static bubble wrap (Blue)	1,805,760.00
2015		**2016**	
StockItemID	**$SalesAmount**	**StockItemID**	**$SalesAmount**
Stk.Air cushion machine (Blue)	3,317,553.00	Stk.Air cushion machine (Blue)	1,441,341.00
Stk.20 mm Double sided bubble wrap 50	2,093,040.00	Stk.10 mm Anti static bubble wrap (Blue)	946,440.00
Stk.32 mm Anti static bubble wrap (Blue)	2,028,600.00	Stk.20 mm Double sided bubble wrap 50	853,200.00
Stk.20 mm Anti static bubble wrap (Blue)	2,013,480.00	Stk.32 mm Anti static bubble wrap (Blue)	750,750.00
Stk.10 mm Anti static bubble wrap (Blue)	2,011,680.00	Stk.32 mm Double sided bubble wrap 50	744,800.00

Table 3. Top 5 products sold per year.

Description of Business Problem and Details of Business Needs

During our first analysis, we found the following focal point: the company wants to expand in other countries. First, we analysed the workflow of the customer orders. Below, a schema (*Figure 2*) that shows how the workflow works:

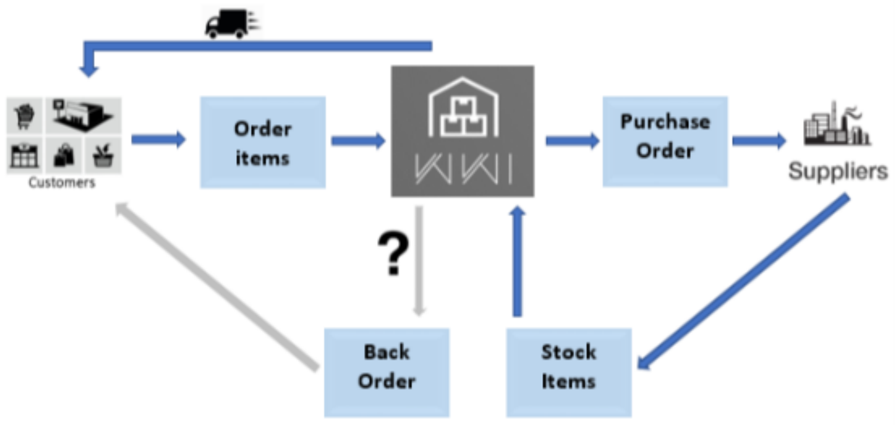


Figure 2. Description of the workflow.

In order to achieve the expansion, the company needs to decrease the backorders and increase the sales. That's why it's crucial to understand how much of our sales were backorders, as well the sales lost due to rejected backorders by the customer.

		Backordered		Sales Lost	
Year	Total Sales	Amount	% of Sales	Amount	% of Sales
2013	$ 45,707,188	$ 3,662,455	8.01%	$ 816,606	1.79%
2014	$ 49,929,487	$ 2,083,781	4.17%	$1,465,573	2.94%
2015	$ 53,991,490	$ 4,662,440	8.64%	$2,014,048	3.73%
2016**	$ 22,633,176	$ 5,173,154	22.86%	$1,076,709	4.76%

**until 2016-05-31

Table 4. Backorder and Lost Sales per year.

The summary above (*Table 4*) shows an increase in sales over the last couple years. The backorders take a significant amount. The increasing sales lost are worrying. Therefore, we concluded that sales, the corresponding backorders and lost sales are the major points to take into account for our DataWarehouse design.

Analysing the database

A Business Intelligence solution is required due to an expansion of WWI in the next years. In fact, WWI plans to expand abroad to increase sales and market share, to ensure this transition it is important to analyse the current recourses available, for example sales employees with different languages, and we will be able to choose strategies to improve our market position and understand the profitability of our business broken down by Customer, Product, Sales etc. Therefore, rejected customer backorders create a loss of potential sales and what impact it has on our sales position.

Considering all this, we imagined which are the possible questions for the business needs:

- Which is the most sold product per day, week, month, quarter and year?
- What are the sales per region, continent, nation, province and town?
- Which is the loss of the potential sales per day, week, month, quarter and year?
- Which is the impact of the backorders on the sales?
- Which is the product with more backorders?
- Which is the period of the year in which we encounter more crucialities?
- Who is the salesperson with the highest sales?

Description of Source Systems and of Source Data

The Source Systems is composed by four complex and big schemas: Applicational, Buying, Selling and Warehousing.
Applicational schema has information about users and their location, contacts, parameters and payment and delivery method.
Buying schema contains details of the purchases from the suppliers and figures about them, for instance, vendor categories and movements.
Selling schema involves details about customers (client categories, special deals buying groups, client movements), item sales, salespeople and invoices.
Warehousing schema stocks item inventory, with their features, and transactions.
In the table below (*Table 5*), we described in a more schematic way the source data in its content, the fields we decided to use and the purpose of them:

Source Systems	Contents	Our Purpose	Main fields
Buying	Stock item purchases from suppliers and details about suppliers	Schema out of our purposes contains information about the purchases from the suppliers	BuyingOrders.SupplierID, BuyingOrderDetails.PurchaseOrderID, VendorCategories.SupplierCategoryName, Vendors.SupplierName
Selling	Stock item sales to retail customers, and details about customers and sales people	Identify the best customers and analyze sales, lost sales and backorders	Clients.CustomerID, ClientVIPCategory, Clients.CategoriesClientCategoryName, Clients.CustomerName, Clients.DeliveryPostalCode, Clients.DeliveryAddressLine1, Clients.DeliveryAddressLine2, Clients.PostalPostalCode, Clients.PostalAddressLine1, Clients.PostalAddressLine2
Applicational	Application-wide users, contacts, and parameters. This also contains reference tables with data that is used by multiple schemas	Identify best sales person and sales regions where they sell more	Nations.Region, Nations.Continent, Nations.Subregion, Nations.NationName, Provinces.SalesTerritory, Provinces.StateProvinceName, Towns.TownName, Person.PersonID, Person.IsEmployee, Person.IsSalesPerson, Person.FullName, Person.OtherLanguages, Person.PhoneNumber

Warehousing	Stock item inventory and transactions	Product with highest sales	InventoryItems.StockItemID, InventoryItems.StockItemName, PackageCategories.PackageTypeName, InventoryItems.Barcode, InventoryItems.Brand, InventoryItems.Size, InventoryItemHoldings.LastCostPrice

Table 5. Description of Source Data.

Description of Staging Area and of the Data Warehouse

A staging area is mainly required in a Data Warehousing Architecture for timing reasons. In short, all required data must be available before data can be integrated into the Data Warehouse. We thought it might be reasonable to extract the data on a daily basis, however, daily extractions might not be suitable for a general and weekly/monthly view of the company. For all these reasons, we chose to use a persistent Staging Area. Instead, regarding the Data Warehouse, we decided to load the data on a weekly basis.

It was taken into consideration to obtain the sales information from Invoices and InvoiceDetails tables, but due to less granularity and lack of information regarding backorders and lost sales, Orders and OrderDetails are more appropriate sources. To have a sophisticated proof we matched the Invoice amounts against the finished orders (sales amount) shown in the following table (*Table 6*). The total of both matches. There are small discrepancies among the years, but they are not of significant interest, as this DataWarehouse will not be used as reconciliation for accounting/audit purposes.

Year	SalesAmount*	InvoiceAmount**
2013	$46,097,907	$45,707,188
2014	$50,026,914	$49,929,487
2015	$53,802,429	$53,991,490
2016	$22,334,091	$22,633,176
Total	**$172,261,341**	**$172,261,341**

*based on OrderDate **based on InvoiceDate

Table 6. SalesAmount and InvoiceAmount per year.

The time passed between when an order was placed to when it became a sale, is an important point to take into account for our Data Warehouse design as it impacts the measures we want to represent. The main issue here was to decide when an order turned into sales/backorder or lost sales because if an order was placed today, it might take 1-3 days to become a sale. To improve the accuracy, we will only load our measures showing the records using a rolling cut-off date[1] which is 7 days in the past, e.g. load data on 30.11.2017 the latest record in the DataWarehouse will be not more recent than 23.11.2017. This way, we decrease the chance of discrepancies between orders, sales, backorders and lost sales. We considered to create a second fact table[2] to represent lost sales by a monthly granularity, but to keep a daily granularity across all measures and as our choice for the Data Warehouse design is a Star Schema, we decided to use a cut-off date for loading data/records.

Moreover, we decided to use Incremental Load and not Full Load. The Full Load read unnecessarily old records that we don't need to be read as we had already processed them before. In our case, we will pay

[1] See ETL processes, Data Warehouse

[2] See Justification of the Star Schema

attention to read those records that are not already loaded. It is for this reason that we used the parameter 'Latest Loaded Date' that will show us the latest date of loading in the Staging Area.
The main advantages we identified for implementing incremental loads are:

- Improving efficiency for ETL process, wouldn't be practical to reload the same records e.g. every night;
- Therefore, fewer resources required, source and destination server downtimes are much shorter;
- Overall it will decrease costs and increase system performance.

We developed the design of the Staging Area and of the Data Warehouse using Microsoft SQL Server Management Studio.

Designing the Staging Area
Below you can find listed the justifications and the descriptions that drove us to build the schema. There is a fact table ("Fact_Sales") and five dimension tables ("Dim_Date", "Dim_Product", "Dim_Location", "Dim_Customer", "Dim_Staff").

Fact Table: "Fact_Sales"
The fact sales table contains 8 measures that aim to answer to our business questions shown here in the following table (*Table 7*). For each business question[3], we created a measure that we can see in the second column.

Business need	Measures
Which is the most ordered product?	OrderQuantity
Which is the profit or loss?	ProfitLossAmount
Which is the impact of the backorders on the sales?	LostSalesAmount
Which is the most sold product?	SalesQuantity, SalesAmount
Which are the products with more backorders?	BackOrderQuantity, LostSalesQuantity
Which is the impact of the backorders on the sales?	LostSalesAmount, BackOrderAmount

Table 7. Requested measures.

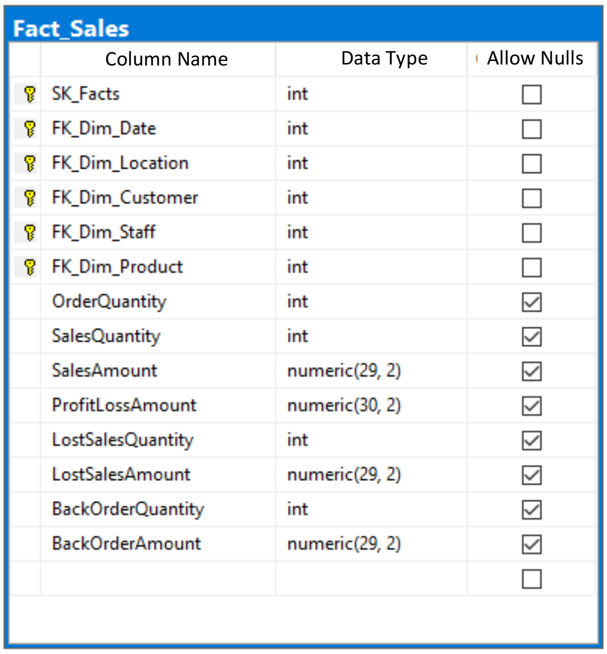
[3] See Analysing the database

We can see a description of measures that we have in our fact table (*Table 8*): from where we took this information about the original table and column in the source system, which is the type of fact, the data type and at the end there is the formula we used to calculate the measure.

Measures	**Table**	**OLTP Fields**	**Type of Fact**	**Data Type**	**Formula****
OrderQuantity	Selling.OrderDetails	sod.Quantity	Additive	int	=sod.Quantity
SalesQuantity	Selling.OrderDetails	sod.Quantity	Additive	int	=sod.Quantity where sod.PickedQuantity == sod.Quantity
SalesAmount	Selling.OrderDetails	sod.Quantity, sod.UnitPrice	Additive	numeric(29, 2)	=sod.Quantity*sod.UnitPrice where sod.PickedQuantity == sod.Quantity
ProfitLossAmount	Selling.OrderDetails Warehousing.InventoryItem Holdings	sod.Quantity, sod.UnitPrice, wiih.LastCostPrice	Additive	numeric(30, 2)	=(sod.Quantity*sod.UnitPrice) - (sod.Quantity*wiih.LastCostPrice) where sordd.PickedQuantity == sordd.Quantity
LostSalesQuantity	Selling.OrderDetails, Selling.Orders	sod.Quantity, sord.BackorderOrderID, sod.PickedQuanity	Additive	int	=sod.Quantity where sod.PickedQuantity == 0 and sord.BackorderOrderID is null
LostSalesAmount	Selling.OrderDetails, Selling.Orders	sod.Quantity, sod.UnitPrice, sord.BackorderOrderID, sod.PickedQuanity	Additive	numeric(29, 2)	=sod.Quantity*sod.UnitPrice where sod.PickedQuanity == 0 and sord.BackorderOrderID is null
BackOrderQuantity	Selling.OrderDetails	sod.Quantity, sord.BackorderOrderID	Additive	int	=sod.Quantity where sord.BackorderOrderID is not null
BackOrderAmount	Selling.OrderDetails	sod.Quantity, sod.UnitPrice, sord.BackorderOrderID,	Additive	numeric(29, 2)	=sod.Quantity*sod.UnitPrice where BackorderOrderID is not null

***sod: Selling.OrderDetails; wiih: Warehousing.InventoryItemHoldings; sord: Selling.Orders*

Table 8. Finalized measures.



Fact_Sales

	Column Name	Data Type	Allow Nulls
🔑	SK_Facts	int	☐
🔑	FK_Dim_Date	int	☐
🔑	FK_Dim_Location	int	☐
🔑	FK_Dim_Customer	int	☐
🔑	FK_Dim_Staff	int	☐
🔑	FK_Dim_Product	int	☐
	OrderQuantity	int	☑
	SalesQuantity	int	☑
	SalesAmount	numeric(29, 2)	☑
	ProfitLossAmount	numeric(30, 2)	☑
	LostSalesQuantity	int	☑
	LostSalesAmount	numeric(29, 2)	☑
	BackOrderQuantity	int	☑
	BackOrderAmount	numeric(29, 2)	☑
			☐

Table 9. Fact Sales.

To conclude, we chose the facts as mentioned in the table above (*Table 7, Table 8*):

- OrderQuantity (quantity orders, based on purchase orders by customer);
- SalesQuantity and SalesAmount (what has been sold based on the orders);
- ProfitLossAmount (profit or loss);
- BackOrderQuantity and BackOrderAmount (backorders due to no stock);
- LostSalesQuantity and LostSalesAmount (rejected backorders).

The fact table contains measures to build a Data Warehouse solution, which fits the purpose of developing a BI solution in the future. We would like to pay attention to the order quantities and sales quantities and to the sales amount in dollars by adding these kinds of measures. Thus, we would like to pay attention to the possible Loss Amount, because of a better understanding of the strategy of the company. Moreover, we want to understand the backorder quantities and amount, and the respective loss of potential sales.
As it will be explained below, we needed to add a surrogate key for the fact table.

Dimensions

To let the business have the sales under control, we would like to understand the sales and corresponding quantities by geographical aspects and date. We would like to be able to analyse the sales and orders for every type of granularity we chose; for instance, we would be able to know the total amount of sales per year and per region. For that reason, we needed a high level of granularity in our dimension tables.
Since it is important to know which one of our customers are the best and which is their potential, we decided to consider a Customer dimension table.
For the same reason, and because it is important to know everything about our products, if costumers like them, which is our best seller products and etc, we considered also a Product dimension table.
Moreover, we added the Staff dimension table to know the skills of the employees of the company.

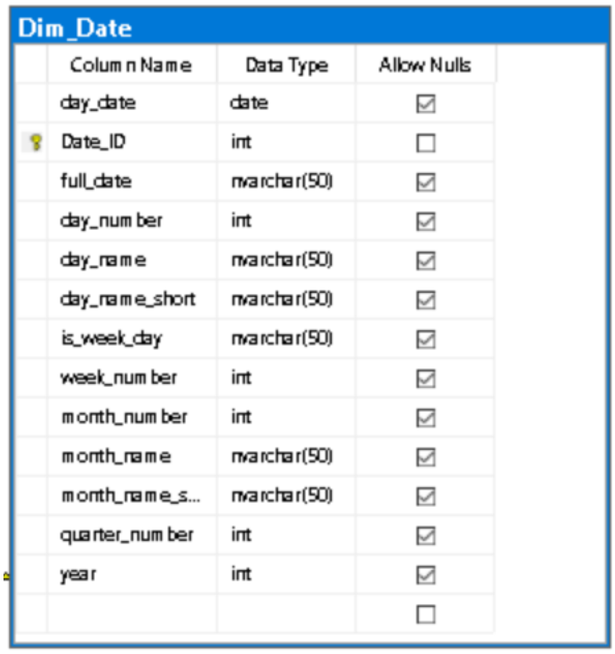
In the following table (*Table 10*), there are the five dimensions that we chose for our staging area, we can also see from which table and which column of the Source System they are extracted, and we can see the data type of the key.

Dimension	OLT Entities	Key Data Type	Parent Dimension
Customer	selling.client, selling.clientcategories	int	None
Staff	Applicational.Person	int	None
Product	Warehousing.InventoryItems, Warehousing.InventoryItemHoldings, Warehousing.StockBunches, Warehousing.packagecategories	int	None
Date	Date csv flat file	int	None
Location	Applicational.Nations, Applicational.Provinces, Applicational.Towns, Applicational.SystemParameters	int	None

Table 10. Requested dimensions.

Let's now see the description of each one of the dimension table.

Dimension Table "Dim_Date"



Dim_Date

Column Name	Data Type	Allow Nulls
day_date	date	☑
Date_ID	int	☐
full_date	nvarchar(50)	☑
day_number	int	☑
day_name	nvarchar(50)	☑
day_name_short	nvarchar(50)	☑
is_week_day	nvarchar(50)	☑
week_number	int	☑
month_number	int	☑
month_name	nvarchar(50)	☑
month_name_s...	nvarchar(50)	☑
quarter_number	int	☑
year	int	☑
		☐

In this table, we have all the information about the date. In here, we can also see the first (year, quarter, month, week, day) of our five hierarchies with five levels of depth. This hierarchy permits a lot of granularity of our fact table.

In this type of tables, Date dimension table, it is important to have a high level of granularity because we should give many ways to aggregate, analyse and report the data. For instance, the company would want to see the evolution of sales during the years, during the quarter, or during the months. In addition, it is also possible to analyse which is the day in which people order more or which is the worse day for the selling.

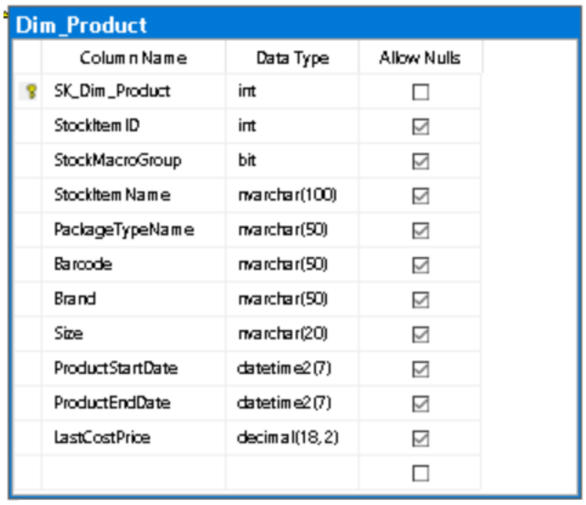
Table 11. Date dimension.

The hierarchy of this table is the following:

Year → Quarter → Month → Week → Day

The Date dimension cannot be populated with the values available in the source systems. As such, we opted to use a csv file. The file we used is: "Date_v.f.csv”. A comma separates the records in this file and there are no commas within the fields; in fact, the csv file is controlled and pre-processed.
Since the first column of this csv file is the Date_ID, an integer that is unique for any record, we did not need a surrogate key, and so we use the Date_ID as the primary key.

Dimension Table “Dim_Product”



Dim_Product

Column Name	Data Type	Allow Nulls
SK_Dim_Product	int	☐
StockItem ID	int	☑
StockMacroGroup	bit	☑
StockItem Name	nvarchar(100)	☑
PackageTypeName	nvarchar(50)	☑
Barcode	nvarchar(50)	☑
Brand	nvarchar(50)	☑
Size	nvarchar(20)	☑
ProductStartDate	datetime2(7)	☑
ProductEndDate	datetime2(7)	☑
LastCostPrice	decimal(18, 2)	☑
		☐

Table 12. Product Dimension.

In this dimension table, we put all the attributes that we reputed useful to get all the information about the product (the name, the barcode, the group, the type of the package, the size, the brand, the last cost price). As a key, we use a surrogate key SK_Dim_Product. This dimension contains the second hierarchy (StockMacroGroup, Brand, Size, StockItemName). The StockMacroGroup divides the products into chilled and non-chilled.
Before drawing the hierarchies, it is important to say that the field StockGroupName is not considered in our hierarchy anymore because we noticed that there could be multiple StockGroupName(s) that belong to each StockItemID, and in this case, a hierarchy is not defined. Therefore, we exclude it from the hierarchy.

The hierarchy of this table is the following:

StockMacroGroup → PackageTypeName → Brand → Size → StockItemName

We also considered another hierarchy, for example, if we are more interested in the Brand, in terms of marketing:

StockMacroGroup → Brand → StockItemName

In this dimension table, we inserted some information about the price of the product. To achieve this objective, we could choose one of the two columns: LastCostPrice or UnitPrice. We decided to use LastCostPrice, as this is a more accurate attribute, shown by lower value than the UnitPrice.
In this dimension table, we used the Slowly Changing Dimension Wizard to manage the types of changes in Dimension attributes. The two attributes that we are going to change are StockItemName and LastCostPrice in the following way:

- StockItemName as changing attribute (Type 1): In this case, the old name will be replaced with the new one and we do not keep track of the old name. This is because we need to know the products we have, and their names and we are not interested in keeping track of all the old, not used, names.
- LastCostPrice as a historical attribute (Type 2): In this case, we want to keep track of all the costs of the products. Normally this kind of change can be done in two ways; we can add a new record with the new cost or, in addition to the new row, we also add two columns to keep track of the StartDate and EndDate. This second way is the one we used. We do that because for us was important to keep track of the LastCostPrice and when it changes.

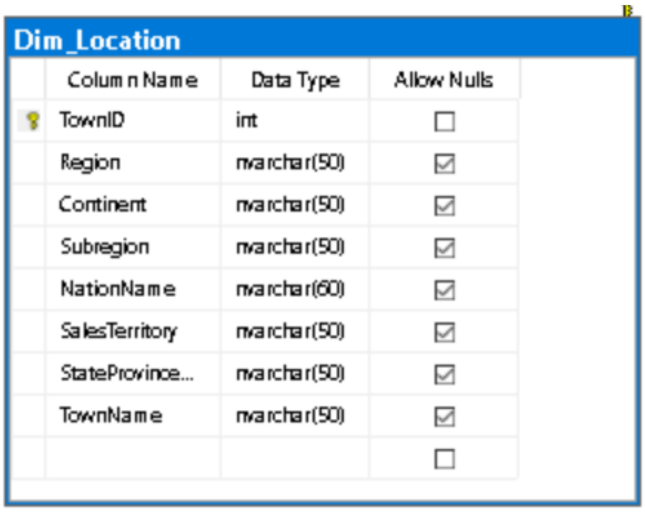
We also decided to use a surrogate key, so loading the Staging Area will be easier and faster.

Dimension Table "Dim_Location"

This table contains all the geographical information related to the customer. As a primary key, we decided to pick the TownID as it is the lowest level of the hierarchy we could use to define an unequivocal correspondence between the customer and its location.
In here there is the third hierarchy formed by Region, Continent, Subregion, NationName, SalesTerritory, StateProvinceTown, TownName.

Region → Continent → Subregion → NationName → SalesTerritory → StateProvinceTown → TownName

It is important to underly that the field PostalPostalCode is no more considered in our hierarchy because we noticed that there were different PostalPostalCode referring to the same TownID and vice versa, a different TownID with the same PostalPostalCode. In this case a hierarchy is not defined. Therefore, we exclude it from the hierarchy.
Since WWI is planning to expand abroad, we believed that could be useful analysing and exploring the location of our customers, to have a better understanding in which geographic areas the business should focus on.



Dim_Location

	Column Name	Data Type	Allow Nulls
🔑	TownID	int	☐
	Region	nvarchar(50)	☑
	Continent	nvarchar(50)	☑
	Subregion	nvarchar(50)	☑
	NationName	nvarchar(60)	☑
	SalesTerritory	nvarchar(50)	☑
	StateProvince...	nvarchar(50)	☑
	TownName	nvarchar(50)	☑
			☐

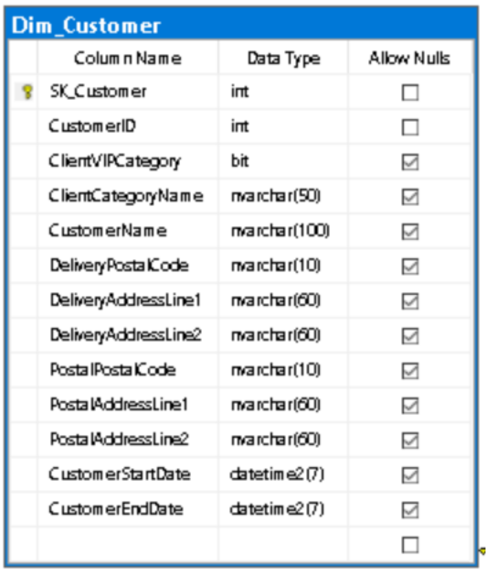
It is common to have a dimension location that connects the geography with other data. This because, we can analyse orders, sales, and so on with aggregation by region, by continent and so on, deciding also which is the area where we should improve the sales, in which areas there are our best customers, etc.
In this dimension table, we used the Slowly Changing Dimension Wizard to make the Incremental Load working for the location. All the attributes are fixed.

Dimension Table "Dim_Customer"

This table defines the customer in all its essential features like name, delivery information and validity. As a key, we use the surrogate key SK_Customer to make the loading of the Staging Area easier and faster. We can find the fourth hierarchy formed by ClientVIPCategory, ClientCategoryName, CustomerName.

ClientVIPCategory → ClientCategoryName → CustomerName

The first columns of the hierarchy, ClientVIPCategory, is a created Boolean field that divides clients into two groups: VipClients, customers to which is applied a percentage discount and all the others.
We used the Slowly Changing Dimension Wizard to manage the Types of changes in Dimension attributes.



Dim_Customer

	Column Name	Data Type	Allow Nulls
🔑	SK_Customer	int	☐
	CustomerID	int	☐
	ClientVIPCategory	bit	☑
	ClientCategoryName	nvarchar(50)	☑
	CustomerName	nvarchar(100)	☑
	DeliveryPostalCode	nvarchar(10)	☑
	DeliveryAddressLine1	nvarchar(60)	☑
	DeliveryAddressLine2	nvarchar(60)	☑
	PostalPostalCode	nvarchar(10)	☑
	PostalAddressLine1	nvarchar(60)	☑
	PostalAddressLine2	nvarchar(60)	☑
	CustomerStartDate	datetime2(7)	☑
	CustomerEndDate	datetime2(7)	☑
			☐

We keep all fields as a historical attribute (Type 2): in this case, we want to keep track of all the changes of the customers.
In this table, we find relevant to have all the possible information about our client, from the name to the postal code.
It is important for us to know if a customer changes, for example, the delivery address or the name; sometimes after marriage, these things happen, and we must keep track of all these changes to better analyse our clients.

Dimension Table "Dim_Staff"

This table identifies the people working in the sales departments, contact information, validity and the languages spoken. As a key, we used the surrogate key SK_Dim_Staff. We thought that the language-spoken attribute might be important from the moment that the business wants to expand in other countries during the next years. We can see the presence of a hierarchy (IsEmployee, IsSalesPerson, IsMultilingual, FullName). IsMultilingual is a created field that says 1 if the Staff person has the knowledge of other languages.

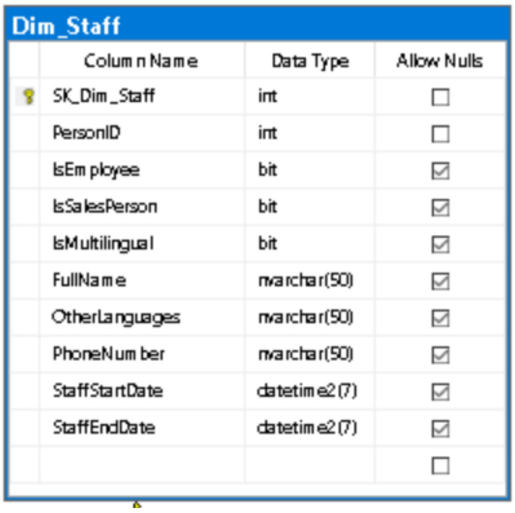
IsEmployee → IsSalesPerson → IsMultilingual → FullName

As happened in the Product dimension table, depending on our focus, we can choose to consider another kind of hierarchy, as, for example:

IsEmployee → IsMultilingual → FullName

If we are interested in knowing if our employee speaks foreigner languages, or if we want to know, for example, which one of my employees are a seller:

IsEmployee → IsSalesPerson → FullName



Dim_Staff

	Column Name	Data Type	Allow Nulls
🔑	SK_Dim_Staff	int	☐
	PersonID	int	☐
	IsEmployee	bit	☑
	IsSalesPerson	bit	☑
	IsMultilingual	bit	☑
	FullName	nvarchar(50)	☑
	OtherLanguages	nvarchar(50)	☑
	PhoneNumber	nvarchar(50)	☑
	StaffStartDate	datetime2(7)	☑
	StaffEndDate	datetime2(7)	☑
			☐

Also in this last dimension table, we used the Slowly Changing Dimension Wizard to manage the Types of changes in Dimension attributes.We keep almost all fields as fixed attributes (Type 0): they don't change. PhoneNumber as changing attribute (Type 1): In this case, the old number will be replaced with the new one. This is because we need to know the newest phone number and we are not interested in keeping track of all the old, not use numbers.

Designing the Data Warehouse

When we started thinking about the design of our DataWarehouse, the first focal point that came up in our minds was the need to take all the surrogate keys from the Fact and Dimension Tables, except for the Customer's one, which it's going to be well explained after .

We proceeded after cleansing some data in each dimension, more in particular:

- **Customer Dimension**: since our business only needs to think about the delivery, we thought that it could be better to take out all the other fields related to the addresses, that the output is clear and simple.
  Another change we made coming from the SA, was trying to differentiate the VipClients from the non-VipClients: in this way, we could better understand to whom the special discounts must be addressed since the VipClients have a positive discount percentage.
  Finally, considering that the identity of the customer sometimes can't change with the surrogate key and that we want to have as much information as possible, we kept both the surrogate key and the primary key.
- **Product Dimension**: since not every product has a specified value for the 'Brand' we agreed about changing this field to let the user understand that the brand for that specified product is not specified. To let the data more readable and user-friendly, we adopted the same strategy for the field 'StockMacrogroup': as it will be well explained in the DataWarehouse's ETL process, we made a change in the data type of the field, to define if the product is chilled or not.
- **Staff Dimension**: for this dimension, we agreed with leaving all the fields related to the person as they were in the StaginArea, since having a description for their fields wouldn't be useful for our purposes;
- **Location and Date Dimensions**: for these two dimensions we decided to leave all the fields as they were in the Staging Area; for Location, we believed that we need to see all the information about the customer's location to better understand the geographic area on which WWI should focus its business strategies; concerning DATE, all the attributes need to be kept to better contextualize the information about the orders.

Justification of the star schema

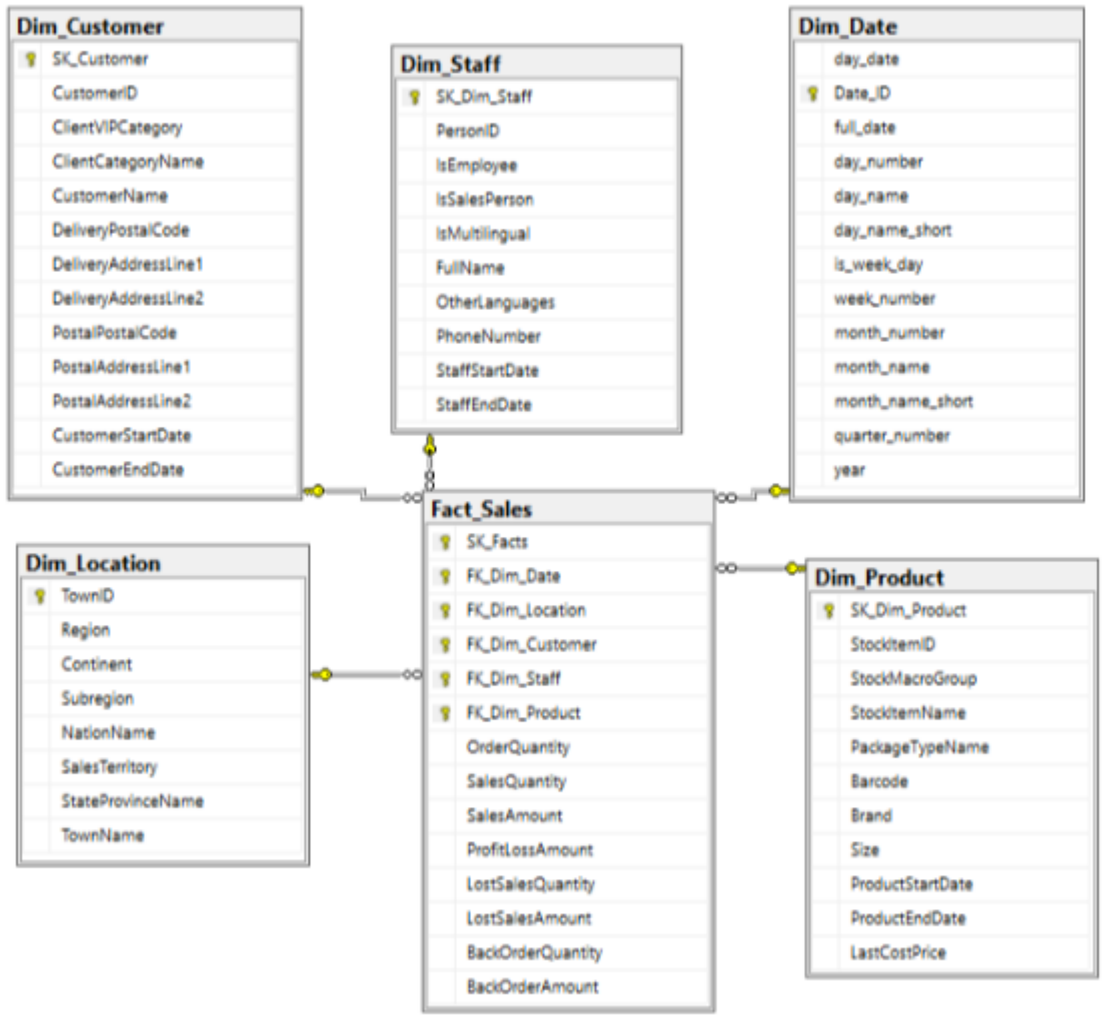
After a first meeting with the business, we have been specifically asked for a Data Warehouse as fast and easy as possible, for the users to get the information easily and by running only simple queries.

A Galaxy Model might have been used to distinguish the quantitative data about sales and orders, but we have agreed that it would have resulted in a more complicated model with normalized dimensions as definition; since we believe that orders can be considered as a part of sales, we decided to adopt a star schema with one fact table and five dimensions.

The main advantages of business are:

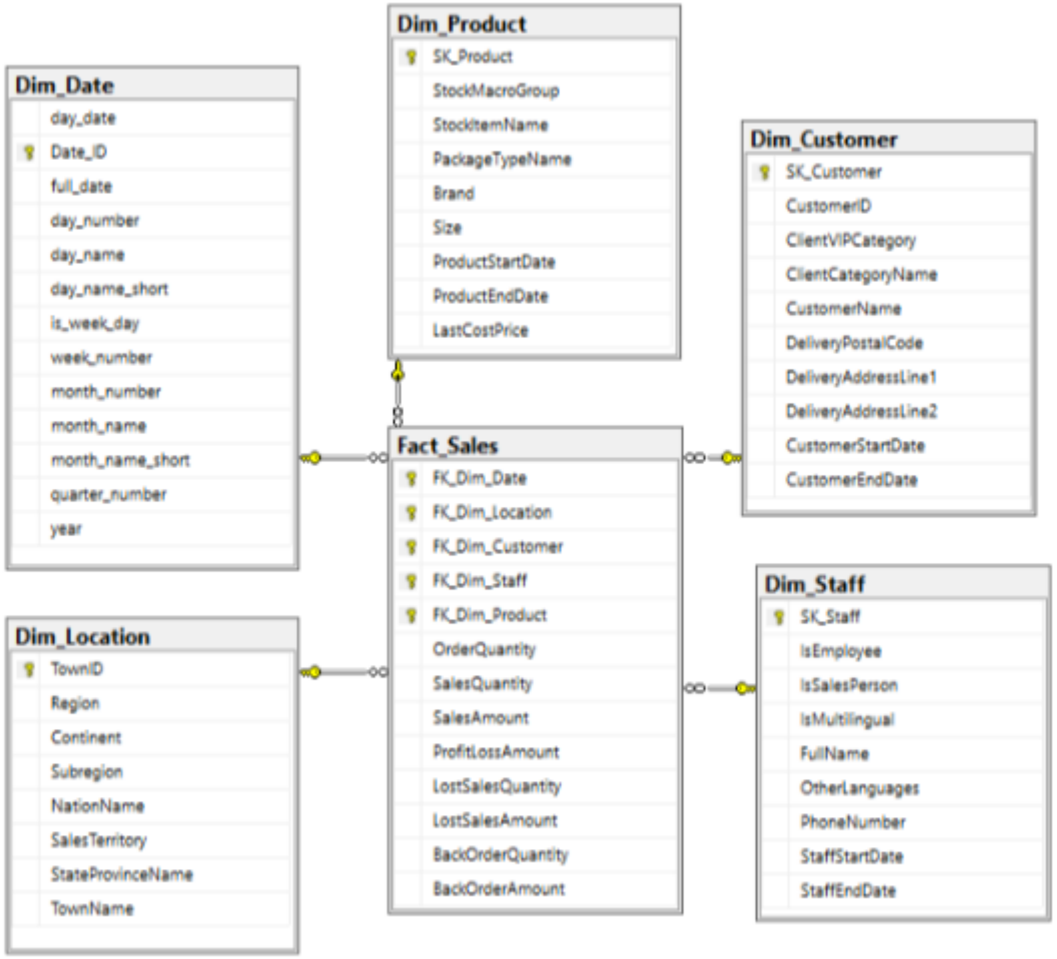
- Query performance: small number of tables and clear join paths, dimensions are only linked through the "central" fact table, therefore higher accuracy and consistent query results;
- Load performance and administration: simple structure ensures less time is required to load data. Moreover, new facts can be added regularly electively by appending records to a fact table;
- Built-in referential integrity, ensured as each record in dimension table has a unique primary key and all keys in fact tables are foreign keys drawn from the dimension tables;
- Easy to understand, better to understand for the end user as they represent relationships of the business.

Staging Area Schema



Schema 1. Staging Area Schema.

Data Warehouse Schema



Schema 2. Data Warehouse Schema.

ETL Processes

Our ETL processes are developed through Microsoft SQL Server Integration Services tool. There are two processes to load in the Staging Area and to load in the Data Warehouse.

Staging Area

Connection Manager

Initially, we added all our connections in "Connection Manager"; one OLE DB connection for WorldWideIMS database, one OLDE DB connection for our Staging Area database and one Flat File Connection for the csv date file.

Control Flow and Data Flow

1. **Truncate Facts**

The first step in our process is to truncate all the fact records. We used truncate because it is suggested when the tables are huge, as a fact table. In fact, "Truncate" is a statement which quickly deletes all records in a table by deallocating the data pages used by the table. Records removed cannot be restored. Another advantage to using it is that in addition to removing all rows from the table it resets the identity back to the seed and the deallocated pages are returned to the system for use in other areas.

2. **Delete Dimensions**

The second step is to delete all the dimension's records. In this case, we used delete for all the five dimensions. The "Delete" statement delete rows one at the time; although this consumes more database resources and locks, these transactions can be rolled back if necessary. Because we use a Star Schema, the order doesn't matter, so we use a Container Sequence for these operations.

3. **Load Dimensions**

For the same reason, we load all the records in the dimensions in a Container Sequence. This step is the preceding of the load facts because before we need the correspondent dimensions.
In the data flow of the load dimensions, we built their table as we described in the design.

3.1. Customer's Dimension

In the load of the costumer's dimension, we created a new column, 'ClientVIPCategory'. Clients with a positive discount, there will be considered "VIP". "In the "Slowly Changing Dimensions" we kept 'CustomerID' as the business key and all the other attributes as historical. In fact, they all can change and we would like to have their changed information (also the 'CustomerName') for future and possible analysis.

3.2. Product's Dimension

In the data flow of "Load Product" we created the column 'StockMacroGroup' (one if it is chill stocked, otherwise zero) and for the slowly changing dimension we selected 'StockItemID' as the business key, 'LastCostPrice' as historical attribute, 'StockItemName' as changing attribute and all the others as fixed. Indeed, we would like to have all the history of the price of the product, but not of the name because we thought the essentiality of the product will not be touched if the name changes.

3.3. Date's Dimension

In the data flow of "Load Date" we needed to convert all the fields from the text type to the appropriate data type. We filtered the date from the csv file; the WorldWideIMS database has records from the year 2013 until 31st May 2016. Moreover, we needed to add the "Slowly Changing Dimensions" considering the incremental load.[4]

3.4. 3.4 Staff's Dimension

In the data flow of "Load Staff" we created the column 'IsMultilingual' as a bit type; 1 if the staff person knows other languages. Moreover, we converted in the appropriate type the column 'OtherLanguages'. In the "Slowly Changing Dimensions" we kept 'PersonID' as business key, 'PhoneNumber' as changing attribute and all the others are historical attributes. We would like to have all the information of the development and

[4] See ETL processes, Incremental Load

improvement of the quality of the staff people, but we would like to have the actual telephone number in the case the company needs this information.

3.5. Location's Dimension

We proceeded with the load of the last dimension: Location. We needed to add the "Slowly Changing Dimensions", as in the dimension of the date, to be able to control the incremental load. Moreover, we would like to detail that we didn't need the command "Merge" and then, "Sort" because we did not need an extra flat file for our location.

4. Load Facts

Finally, we loaded our facts. We paid attention to add all the measures with the appropriate data type. Moreover, as we explained[5], we considered the 'OrderDate' as the date key; for this reason, we paid attention to convert it as an integer. In addition, we controlled through five "look up's" the business keys and the surrogate keys for all the dimensions; we loaded only the "active" ones adding a where clause. The "look up" part was crucial for us; we discover that our "Load Product" was wrong; the query was not correct (as we used a cross join in the query). After the appropriate modify, the "look up" had the expected results.
Proceeding with the loading, we found out another problem; we had 14 rows with the same 'FK_Dim_Location', 'FK_Dim_Date', 'FK_Dim_Product', 'FK_Dim_Customer', 'FK_Dim_Staff' of other 14 rows (*Table 16*).

FK_Dim_ Product*	FK_Dim_Date*	Order Quantity	Sales Quantity	Sales Amount	LostSales Quantity	ProfitLoss Amount	LostSales Quantity	LostSales Amount	BackOrder Quantity	BackOrder Amount
35561	20140909	9	9	288	0	180	0	0	0	0
35561	20140909	3	3	96	0	60	0	0	0	0
35565	20140929	2	2	64	0	40	0	0	0	0
35565	20140929	7	7	224	0	140	0	0	0	0
35584	20140221	1	1	13	0	8.5	0	0	0	0
35584	20140221	9	9	117	0	76.5	0	0	0	0
35585	20141011	5	5	65	0	42.5	0	0	0	0
35585	20141011	5	5	65	0	42.5	0	0	0	0
35589	20160331	1	1	13	0	8.5	0	0	1	13
35589	20160331	9	9	117	0	76.5	0	0	0	0
35592	20160526	1	1	13	0	8.5	0	0	0	0
35592	20160526	3	3	39	0	25.5	0	0	3	39
35605	20160411	9	9	117	0	76.5	0	0	0	0
35605	20160411	10	10	130	0	85	0	0	0	0
35611	20150727	2	2	50	0	25	0	0	0	0
35611	20150727	10	10	250	0	125	0	0	0	0
35630	20140313	120	120	2160	0	1320	0	0	120	2160
35630	20140313	60	60	1080	0	660	0	0	0	0
35636	20150407	72	72	1296	0	720	0	0	0	0
35636	20150407	108	108	1944	0	1080	0	0	0	0
35639	20140926	120	120	2160	0	1140	0	0	0	0
35639	20140926	24	24	432	0	228	0	0	0	0
35668	20140509	5	5	170	0	50	0	0	0	0
35668	20140509	6	6	204	0	60	0	0	0	0
35737	20160322	175	175	367.5	0	175	0	0	0	0
35737	20160322	250	250	525	0	250	0	0	0	0
35747	20140508	192	192	787.2	0	403.2	0	0	0	0
35747	20140508	144	144	590.4	0	302.4	0	0	0	0

**only two foreign keys to make the table readable.*

Table 16. Showing the problematic 28 rows.

[5] See Description of the Staging Area and Data Warehouse

We studied how it could happen. We noticed that two of the 28 problematic rows are exactly the same so it could be a duplication's problem in the data source, but there wasn't sufficient proof that this is a case. For that reason, we thought it is a problem of the granularity of the date we chose. In fact, a customer can order twice or more on the same date but our date considers only the date and not the granularity of the time. In fact, considering the 'OrderLineID', we can see they are all different and there is not the problem of uniqueness.
To solve the problem, we added a surrogate key in the fact table; we researched on the web about this decision if it was possible and common. Usually, it is preferable to not do it, but we considered this one as the best solution for the staging area, to have a fast and easier loading. In the Data Warehouse, we solved the problem in another way, the loading is less efficient in term of time but more accurate.
The last problem we occurred in this phase was the number of records, out from the "DefaultBufferMaxRows". We saw that the number of rows in 'OrderDetails' is 231'412 and our results were coherent. We needed to change "AutoAdjustBufferSize" in true and "DefaultBufferMaxRows" bigger than 231'412.

Incremental Load
As we explained before, it is essential to have an incremental load. If it is incremental, we created an option to go directly in the load of the dimensions; if it is not, the process will start passing from the "Truncate Facts". To control it, we added a parameter called 'LatestOrderDate'.

Data Warehouse

The Data Warehouse is a database which represents the final step of our project development (*Figure 3*) and it's where all the data processed in the staging area and previously loaded from the OLTP Database are stored; here the users can execute the queries and extract the historical information they need, in order to analyse the facts and answer the questions on which the entire structure has been built on.

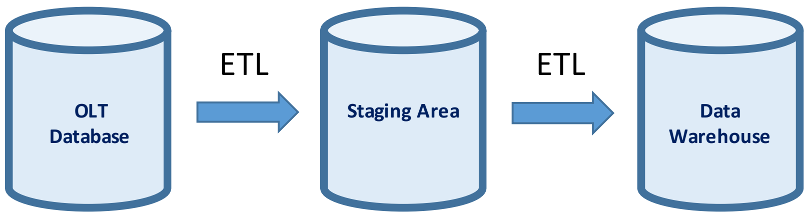


Figure 3. DW process description.

Connection manager
To facilitate the connection between Staging Area and Data Warehouse, we converted in Connection Manager the connection to the Staging Area in a project connection.

Control Flow and Data Flow
In the Data Warehouse we don't have the Truncate for the Fact Table and Delete for the dimensions because, as the DataWarehouse has to be a warehouse, we want to keep in it the data that have been previously loaded from the staging area.

1. Load Dimensions
As we did for the staging area, we load all the dimensions in the Container Sequence called "Load Dimensions"; this step comes before the "Load Facts" because to load them, we need the following dimensions

1.1. Product's Dimension

What we did differently from the staging area in this process, is changing the 'StockMacroGroup': in the staging area this field had the data type bit, which means that it could be 0 or 1; in the data warehouse we wanted to make the results more understandable and readable for possible studies regarding the company, therefore we believed that a description for this field was needed: we will have 'ChilledFood' if the field has the value 1 and 'NotChilled' if the field is valuated with 0.

For what about concern the "Slowly Changing Dimension", we put the 'SK_Product' as a Business Key and then we don't consider it to be any more a surrogate identity, with 1 as a number to be incremented; all the other fields are fixed except for the 'StockItemName' that is changing in order to be coherent with the Staging Area and for the 'EndDate', which would compromise the incremental load for the DataWarehouse if set to fixed.

We've also made a change to the brand field: since not all the products have a specified value in the field 'Brand', we believed that it would be useful to change the values from "Null" to "Unknown".

1.2. Customer's Dimension

In this task, to let the data about 'ClientVipCategory' fields more understandable, we changed its datatype from bit to a string in order to have more information about it: the new values will be VipClient if the field was previously valued with 1 and Client if the field had 0. As we mentioned before, being VipClient means having a discount percentage greater than zero.

We decided to keep both the 'SK_Customer' and the 'CustomerID' because the identity of the customer sometimes can't change with the surrogate key and we would like to have more information to analyse more accurately.

'PostalPostalCode', 'AddressLine1', 'AddressLine2' have been deleted because since we only care about the delivery, we don't need those; moreover, until now PostalCode and DeliveryPostalCode resulted to be the same.

For what about concern the "Slowly Changing Dimension", we put the 'SK_Customer' as a business key since is not anymore, an "identity" with the meaning of auto-increment field, but it's still an identity as it is the surrogate key; then we put 'EndDate' as a changing attribute otherwise there wouldn't be an incremental load for the DataWarehouse; all the other attributes are fixed.

1.3. Staff's Dimension

In this task, we left bit for all the three fields with that datatype ('IsSalesPerson', 'IsMultilingual', 'IsEmployee') because we believed that, having a description for these fields, wouldn't be useful for our purposes.

In the SCD, all the attributes are fixed except for 'EndDate' which is changing in order not to violate the incremental load of the DataWarehouse's process.

1.4. Location's Dimension

In this task, we didn't do anything different from the Staging Area and also here we put TownID as a BusinessKey during the SCD.

1.5. Date's Dimension

In this task, we didn't do anything different from the StagingArea except for putting DateID as a BusinessKey during the SCD.

2. Load Facts

In the DataWarehouse, to solve the problem about the 14 duplicated record lines mentioned above in Staging Area about the duplicate records, we did a research and we found out that the solution, in this case, would

be the use of an aggregate function (Common.Aggregate)[6]: in order to do it, we grouped by the five FK and we summed up the measures; the aggregate function will sum up the 28 records, two per time and it will group them by the five FKs. Doing this, the process will result slower because of the statement “GROUP BY”, but we will not have the 14 extra records as before; therefore, we won’t need the SK in the Fact table anymore.
After we implemented a “Data Conversion” because we encountered a problem with the datatypes of the quantities. We believe that with the sum of the values, they became bigger integers. The new data type for the field is “four-byte signed integer”.
In the “Slowly Changing Dimension” we put all the attributes as changing for us to be able to modify in case of errors, and all the FK as business keys. We decided not to check the box of the option that changes all the matching records, including outdated records, when changes are detecting in a changing attribute: we believe that it couldn’t be possible to have a combination of the five FK at the same time.
Finally, we checked that the number of the rows in the Warehouse was the number of the rows in the Staging Area minus the fourteen extra rows, as a result of the aggregation.
Finally, with a reference to the paragraph where we describe the Staging Area and the Data warehouse, it’s important to remark that we created a new variable ‘CutOffDate’ to solve the problem about loading the appropriate values for our measures, based on interval we determined when an order becomes a sale, what has been backordered or rejected.

Logging
Logging tasks are important because to give insights of the StagingArea and
DataWarehouse developers, operators, system administrators and security team. It is possible to see which ones we use in the following part.

a) In the Staging Area

- “Store Latest Order Date” and “Log Latest Date”

We created a new variable called ‘LatestOrderdate’ as a datetime; this new variable represent the latest date date we have in the Fact table. This logging part is important to control the last order date which the previous process stored and to start the present process from there.

- “Log Incremental Load decision”

It is crucial for the incremental load decision. From here, two possible paths can be followed.
Moreover, it was important to change the property “LogicalAnd” to false, to allow the flow to continue when just one of the row arrives at the end.

- “Log Truncate Facts”, “Log Delete Dimensions”

To check respectively the truncation of the sales facts and the cancellations of the dimensions. and the loading of dimensions and facts.

- “Log Latest Date No Incremental” and “Log Latest Date No Incremental”

Respectively as “Store Latest Order Date” and “Log Latest date”. This logging part controls the last order date when the process is not incremental load.

- “Log Number of Facts Loaded”

We created a new variable called ‘FactsRowCount’ which store the number of loaded rows in the data flow of the “Load Sale Facts”, using a “Multicast”. In the logging part, we wanted to control this number.

[6] https://docs.microsoft.com/en-us/sql/t-sql/functions/aggregate-functions-transact-sql

b) In the Staging Area and in the Data Warehouse

- “Log the start of ETL tasks”, “Log ETL End”

We created a new variable called ‘ETLName’ as a string which contains the datetime (for instance: ETL NAME 31/12/2017 09:23:12). “Log the start of ETL tasks” logs the start of the ETL process. “Log ETL End” logs the end of it.

- “Log Errors”

In “Event Handlers” we store the possible errors in the “Errors_Log” table. We considered “Error” as a description of the error and “Source" as the name of the tasks which is having an error.

- “Log Load Dimensions”, “Log Load Facts”

To check respectively the loading of dimensions and of the facts.

Parameters

Project’s Parameter

To make easier the connection manager with the flat file, we created a parameter ‘InputFilePath’ in which users can specify the path to the date csv file (Nb. Adding the "\").

Package’s Parameter

In the Staging Area, we created a parameter ‘ParamIncrementalLoad’; if it ‘Yes’ (Nb. Case sensitive) then it will execute the incremental load if it ‘No’ then it will not execute it.

Testing

The purpose of testing is to ensure that the implementations work in the appropriate manner. These include the Control Flows, Data Flows, SQL queries and related variables, parameters.
The Data Flow task encapsulates the data flow engine that moves data between sources and destinations, and let us transform, clean, and modify data as it is moving. Addition of a Data Flow task to a package control flow makes it possible for the package to extract, transform, and load data. After several runs of the Staging Area and the Data Warehouse, we can ensure that both Data and Control Flows working as expected, where no further errors were encountered.
For the more complex tasks we tested the Control and Data flow more intensively:

- Staging Area (*Table 17*):

Task	Description	Result	Status
Incremental Load	Adding a dummy record into Selling.OrderDetails and Selling.Orders: OrderDate = 2017-01-01 with OrderID = “99999” OrderLineID = “999999”; Setting ParamIncrementalLoad =”Yes”;	adding 1 new record to Fact_Sales	passed
SCD Load Customer	modifying Selling.Clients;CustomerName = "TESTName"	adding a new record for the same CustomerID, new CustomerName	passed
SCD Load Product	Warehousing.InventoryItems: StockItemName = "Test" Warehousing.InventoryItemsHolding: LastCostPrice = 999.0; different StockItemID's	new record for the LastCostPrice; StockItemName for a record changed to "Test"	passed
SCD Load Staff	Application.Person: FullName = "TestStaff" and PhoneNumber = 99999999	new record for "TestStaff" and changed PhoneNumber to 99999999	passed
SCD Load Location	Application.Towns: TownName = "TestTown"	no change or new record (fixed)	passed
Load Facts: CutOffDate	adding a new recordSelling.Order and Selling.OrderDetails '2018-01-06'; execute ETL's for SA and DW	new record loaded into SA, no new record loaded into DW	passed
Aggregate	Checking records and amounts loaded from SA and DW; the 14 "duplicated" rows should now be aggregated	SA: 231412 records DW 231398 records, measures *amount, *quantity match	passed

Table 17. Testing the Staging Area.

- Instead, for the Data Warehouse (*Table 18*)

Task	Description	Result	Status
Load Facts: CutOffDate	adding a new recordSelling.Order and Selling.OrderDetails '2018-01-06'; execute ETL's for SA and DW	new record loaded into SA, no new record loaded into DW	passed
Aggregate	Checking records and amounts loaded from SA and DW; the 14 "duplicated" rows should now be aggregated	SA: 231412 records DW 231398 records, measures *amount, *quantity match	passed

Table 18. Testing the Data Warehouse.

Validation

The purpose of data validation is to ensure that the data loaded into Staging Area and Data Warehouse is complete and is reconciled with the source data (*Table 19*):

Facts Sales	Staging Area	Data Warehouse	Source	Validation
Records	231,412	231,398	231,412	valid
OrderQuantity	9,310,904	9,310,904	9,310,904	valid
SalesQuantity	8,950,628	8,950,628	8,950,628	valid
SalesAmount	$ 172,261,341.20	$ 172,261,341.20	$ 172,261,341.00	valid
ProfitLossAmount	$ 85,729,180.90	$ 85,729,180.90	$ 85,729,180.90	valid
LostSalesQuantity	360,276	360,276	360,276	valid
LostSalesAmount	$ 5,372,935.20	$ 5,372,935.20	$ 5,372,935.20	valid
BackOrderQuantity	806,729	806,729	806,729	valid
BackOrderAmount	$ 15,581,830.35	$ 15,581,830.35	$ 15,581,830.35	valid

Table 19. Validation of the Fact table in the Staging Area and in the Data Warehouse.

The following table shows the validation of the dimension columns. We checked if they appropriate names/ descriptions and values (*Table 20*):

Check: columns have appropriate values	Staging Area	Data Warehouse
Dim_Customer	TRUE	TRUE
Dim_Date	TRUE	TRUE
Dim_Location	TRUE	TRUE
Dim_Staff	TRUE	TRUE
Dim_Product	TRUE	TRUE

Table 20. Validation of the dimensions of the Staging Area and of the Data Warehouse.

The testing and validation of the final solution were proceeded by 2 group members independently to ensure consistency and accuracy.

Business Intelligence recommendation

Before we want to conclude this Data Warehouse design report we would like to recommend Microsoft Power BI as a Business Intelligence solution. It is a suite of business analytics tools that deliver insights throughout the organization. It provides interactive visualizations with self-service business intelligence capabilities, the user-friendly interface and several advantages throughout the software make it an easily implemented BI tool for WWI. The Data Warehouse can be directly imported as SQL server database and used to create interactive and useful dashboards for analysis. The following illustration shows a sample dashboard using our Data Warehouse as source data (Figure 4):

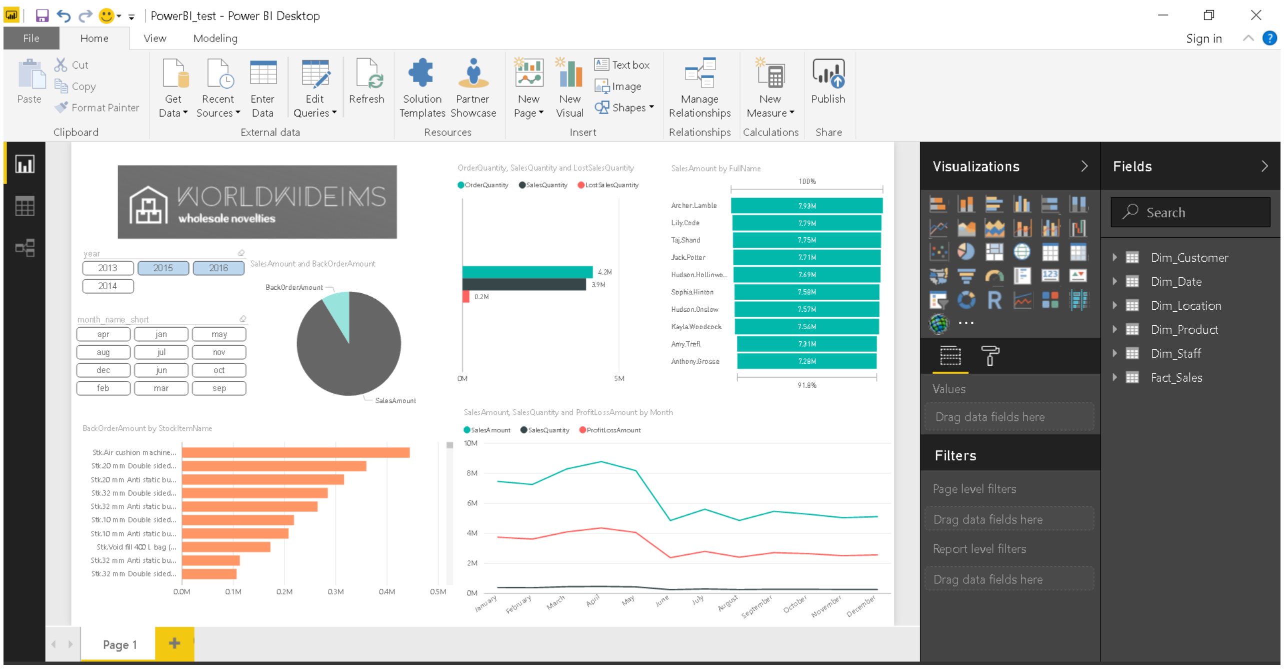


Figure 4. PowerBI dashboard.

Conclusion

The DataWarehouse Design for WorldWideIMS started with the examination of the business including financial statistics and the identification of the business need for a data warehouse. Based on this business requirements we were able to derive questions for building the fundamentals for the further design development. This was accomplished by the description of the source systems and data to interpret the content and purpose of the source to determine the analysis parameters.

The result was a Staging area using incremental load for efficiency and performance reasons build on the principles chosen for the fact and dimension tables. To arrive at an appropriate fact table we based our questions on the business needs to translate them into appropriate measures. This information was used to create a detailed description of the measures and formulas to ensure a consistent workflow throughout the project. Most importantly to understand if the source data matches the expectations for the Data Warehouse. Here we also considered the option to choose between InvoiceDetails and OrderDetails tables from the source data, reconciling the sales and invoice amounts. Further, we needed to understand when an order becomes a sale or was rejected by the customer. Therefore, we had the idea to use a CutOffDate to ensure high accuracy of the records and records loaded. We decided to use the OrderDetails as it fits better into the DataWarehouse requirements. This was followed by the choice of the appropriate dimension and their granularity. Here it was important to choose the hierarchies and make decisions about the slowly changing dimensions. Taking different scenarios and factors into consideration the dimensions and fact tables could be created and therefore our decision was to use the star schema as Data Warehouse design. The decision was based on the simplicity, integrity and performance of this data warehouse schema.

We proceeded with the ETL of the Staging Area coming across different problematics with the source data, for example, we had 14 records with the same foreign keys. The workaround for this issue was to create a surrogated key in the facts table. After the Staging Area, we implemented the Data Warehouse ETL process and took into consideration several things we would like to do in a more appropriate way. This included the aggregation of the records grouped by the FKs to eliminate the SK. We also decided to reduce the dimension tables by excluding columns with no meaningful values. This was concluded by a Logging part for both, Staging Area and Data Warehouse. The Final solution had to be tested and validated by two independent group members, after resolving all the issues we can confirm that the Staging Area, Data Warehouse corresponding records populated are valid and work properly.

To go a step ahead, we thought to include a recommendation for BI solution. We used the Data Warehouse as a source data to build a dashboard in Power BI, also as a proof that this Data Warehouse populates the meaningful and valuable information for the business as expected.