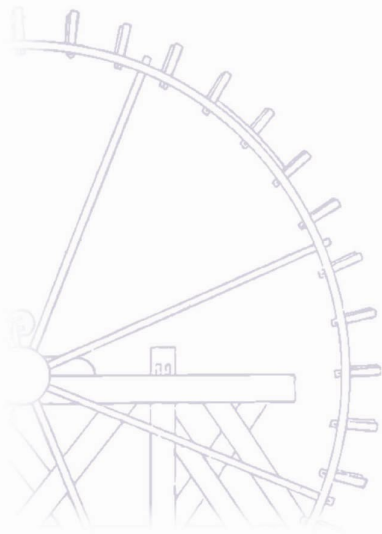


Sualem Rennequin Sualem

INFORMATIQUE

SERAING

BUSINESS INTELLIGENCE WITH I.B.M. DATA WAREHOUSE ENTERPRISE EDITION.

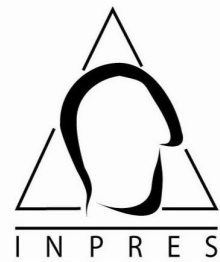


Cédric Namotte

Year : 2006 - 2007

Work presented to obtain the title of bachelor in computer and management sciences

HAUTE ÉCOLE DE LA PROVINCE DE LIÈGE



THANKS

I would like to thanks:

Mister Dimitris Dervos for his accompaniments and his encouragement during the 3 months. More than the work aspect, I met a gentle person with a very good sense of hospitality .

All teachers of the Technical College of Rennequin Sualem for their teaching and their advices. Specially Madam Laurence Herbiet for her precious help and assistance and Mister Joseph Barsics who made all this exceptional training period possible.

My parents and my girlfriend for their daily motivation and presence.

My friend Damien Lejeune for the 3 months that we have spend together in Greece.

Table of Contents

Thanks.....	3
1 Introduction.....	7
2 General context.....	8
2.1 Presentation of the A.T.E.I.....	8
2.2 Goals of the project.....	9
2.3 Technical Environment and used tools	9
3 Presentation of THE IBM Solution.....	10
3.1 DB2 Data Warehouse Edition V9.1.1 overview.....	10
3.1.1 Introduction.....	10
3.1.2 Provided products.....	10
3.2 DB2 DWE architecture.....	11
3.3 DB2 DWE components.....	12
4 The business scenario.....	14
4.1 The fictional company.....	14
4.1.1 Presentation of the JK Superstore.....	14
4.1.2 Understanding of the need of the company.....	14
4.2 The sample database.....	15
4.2.1 The current technical situation of the company.....	15
4.2.2 The Data Warehouse Schema.....	15
4.2.3 The Data Marts Schema.....	17
4.2.3.1 What is a dimensional model.....	17
4.2.3.2 The JK-Superstore Marts.....	18
4.3 Prerequisites before starting with tutorials.....	19
4.3.1 How to install DB2 DWE Edition 9.1.1.....	19
4.3.2 Creation of the database.....	20
5 Design Studio tutorials.....	22
5.1 The Eclipse based Integrated Development Environment.....	22
5.1.1 Introduction to the Design.....	22
5.1.2 Projects in Design Studio.....	22
5.2 Global aim of these tutorials.....	23
5.3 The tutorials and their goals.....	23
6 The physical data model.....	24
6.1 Introduction and prerequisites.....	24
6.2 Creation of the data design project.....	25
6.3 Reverse Engineering the existing database.....	26
6.4 Working on the physical data model.....	30
6.4.1 Defining the problems.....	30
6.4.2 Adding Primary keys.....	31
6.4.3 Adding Associations – Foreign keys.....	31
6.5 Validation of the model.....	33

6.6 Propagation of the modification on the database.....	34
6.6.1 The Comparison process.....	34
6.6.2 The DDL script.....	35
7 The data flows.....	36
7.1 The data warehouse project.....	36
7.1.1 Introduction.....	36
7.1.2 What provides this project?.....	36
7.2 Definition of a data flow in Design Studio.....	37
7.3 Definition of some data flow operators.....	38
7.3.1 File Import operator.....	38
7.3.2 File Export operator.....	38
7.3.3 Distinct operator.....	39
7.3.4 Union operator.....	39
7.3.5 Table Join operator.....	40
7.3.6 Select List operator and Where Condition operator.....	40
7.3.7 Key Lookup operator.....	41
7.3.8 Easy to understand operators.....	41
7.4 Data flows in the JK Superstore.....	42
7.5 Populating the Item Transaction table.....	43
7.5.1 Origin of the data and start point of the flow.....	43
7.5.2 Cleaning the data.....	45
7.6 Populating a dimensional table of the marts schema.....	48
7.6.1 Origin of the data.....	48
7.6.2 Explanation of the problem.....	48
7.6.3 Resolving it within an SQL query.....	49
7.6.4 Designing the first level.....	50
7.6.5 Designing the other levels.....	51
7.6.6 Overview of the data flow.....	52
7.7 Populating a fact table.....	54
7.7.1 Origin and details of the data.....	54
7.7.2 Using the Key lookup operator.....	57
7.7.3 Overview of the flow.....	58
8 Control flows.....	59
8.1 Definition of a control flow in Design Studio.....	59
8.2 Rules and operators from control flows.....	60
8.2.1 Rules.....	60
8.2.2 Outputs.....	60
8.2.3 Definition of available control flow operators.....	61
8.2.3.1 Start operator.....	61
8.2.3.2 End operator.....	61
8.2.3.3 Fail operator.....	61
8.2.3.4 Data Flow operator.....	62
8.2.3.5 File Wait operator.....	62
8.2.3.6 Stored Procedure operator.....	62
8.2.3.7 Variable Comparison operator.....	63

8.2.3.8 Iterative processing loop.....	63
8.2.3.9 Email operator.....	64
8.3 Using the control flow to feed the MARTS schema.....	65
8.3.1 Process understanding.....	65
8.3.2 Designing the data flows.....	66
9 Data warehousing applications.....	68
9.1 Introduction.....	68
9.2 Deployment preparation.....	69
9.3 Working with DWE Administrating console.....	70
9.3.1 Prerequisites before deploying the application.....	70
9.3.2 Deploying the application.....	71
9.4 JK Superstore builds the Data warehouse.....	72
10 Data mining.....	73
10.1 What is the data mining?.....	73
10.2 Why using data mining?.....	73
10.3 Association rules.....	74
10.4 Easy mining procedures.....	76
10.4.1 Introduction to Easy Mining.....	76
10.4.2 FindRules.....	77
10.4.3 ClusterTable	77
10.4.4 The others functions.....	78
10.5 Association rules on the Market Basket of JK Superstore.....	79
10.5.1 Introduction.....	79
10.5.2 Cleaning the data and Creating a model.....	80
10.5.3 The visualizer.....	82
10.5.4 Interpreting the result of the data mining.....	82
10.6 Clustering on the titanic data set.....	88
10.6.1 Introduction.....	88
10.6.2 The origin of the data.....	88
10.6.3 How to create a model without mining flow.....	89
10.6.4 Interpretation of the results.....	90
10.6.4.1 The results with the geometric algorithm.....	90
11 PRESENTATION OF MY PROJECT IN GREECE.....	93
12 ENCOUNTERED AND RESOLVED PROBLEMS.....	94
13 CONCLUSION.....	95
14 BIBLIOGRAPHY.....	96
15 APPENDICES.....	97
15.1 The Java bean code to create the data mining model of titanic data set.....	97

1 INTRODUCTION

To carry out a training period in a foreign country with the Erasmus project was my desire. Discovering new culture, working with foreign persons and improving my English speaking was the advantages of doing a project out of Belgium. My Erasmus training period has been done in Thessaloniki in Greece.

This document present the theoretical notions of Business Intelligence within I.B.M. Data Warehouse Enterprise Edition. These notions will be explained and showed with an end to end example.

First, We will present the DB2 Data Warehouse Enterprise Edition by describing most components like the Intelligent Miner and the Websphere application server.

Next to the presentation, we create a sample database which will be used for the tutorials. This database is the one of JK Superstore, a fictive company. The created database will be then used in Design Studio and loaded into a Data Model Project (OLAP).

After Working on the Data Model Project, we will build a data warehousing application that will be deployed on the Websphere Application server. This application will be designed into a Data warehouse project in Design Studio. The data warehouse project contains data flows, control flows and mining flows.

Finally, we will do data mining on the JK Superstore to try to improve the sales of the company. We will also try data mining on the Titanic data set and try to regroup the population in clusters and check if the rule “Children and women before” is respected.

In the end of the document, I will speak about the presentation I have done at the University of Thessaloniki.

I wish you an excellent reading of this report.

2 GENERAL CONTEXT

2.1 Presentation of the A.T.E.I



*Illustration 1:
A.T.E.I. of
Thessaloniki*

The Alexander T.E.I. of Thessaloniki was established in 1983 under Parliamentary Act 1404/83 "Structure and function of Technological Educational Institutes TEI". It has a student population of approximately 25.000. As far as the number of student is concerned, it holds the fourth position among the Greek Educational Establishments. The Technological Educational Institute aims at providing education of a high quality standard on Applied Sciences and Technologies. Technology transfer from academia and research to real life applications is the sought objective. Curricula in Technological Educational Institutes are structured to meet the needs of the Industry including a 6 months practical training period. Through its Carrier Office TEI of Thessaloniki has developed strong relations with SMEs in the area of Northern Greece.

TEI has developed e-Learning courses for international use, such as Strategic Internet Marketing, AI in logic programming and LAN management. TEI is also engaged in the use of educational standards in relation to development of studies and in research regarding the use of AI techniques, and intelligent agents in relation to e-Learning and Blended learning approaches. By participating in several European projects, e.g. DoODL (1996-98), EuroCompetence (1998-2000), MENU (2001-2003), DBTech (2002-2004) etc., TEI has gained experience in the development of models for virtual universities. Members of TEI staff participating in the project proposal have both business and ICT studies background and combine academic (research) qualifications with professional experience, related to the field of ICT/e-learning approaches for incorporate training.

2.2 Goals of the project

The schedule of conditions has been growing during all the project until the last week. At each appointment with my mentor, we decided together in which direction the project should go depending on what we discovered. During the 3 months, I have been very independent but here is a list of constraints and recommendations, my mentor asks me to reach.

1. An overview of The IBM DWE v.9.1 approach to Business Data / Intelligence: - (What's out there - s/w components, with a brief description for each one of them).
2. For (2): identify the s/w systems/components that you will be working with. Also, identify the relevant IBM literature (a dynamic list which is going to be continuously updated in the course of the project).
3. Writing tutorials that explains step by step how to built a data warehousing application.
4. When the data warehousing is built, try out the Intelligent Miner.

2.3 Technical Environment and used tools

The DB2 Data Warehouse Edition was installed on my laptop. In the other hand, I have installed also the complete solution at the university of Thessaloniki and on the laptop of my room mate (Linux) so that I was able to work over Ethernet and not limiting my work on a local machine. IBM DB2 Control center were installed to manage the servers.

I encounter some problems when the database and the application server were running together on my laptop due to a lake of memory.

3 PRESENTATION OF THE IBM SOLUTION



3.1 DB2 Data Warehouse Edition V9.1.1 overview

3.1.1 Introduction

DB2 Data warehouse edition (DWE) is a compilation of softwares. DB2 DWE uses the famous I.B.M DB2 Relational Database Management System. DB2 DWE is a combination of the DB2 Universal Database with a business intelligence infrastructure.

3.1.2 Provided products

Here is a list of the products included in DWE Professional Edition:

- Design Studio
- DWE SQL Warehousing tool
- DWE Administration console
- DB2 Server Edition
- DB2 Cube View
- DB2 Query Patroller
- DWE Intelligent Miner
- DB2 Alphablox
- Websphere application server

3.2 DB2 DWE architecture

The architecture of DB2 DWE is client-server. In fact, there are three groups of components. Each component can be installed on different computers of a same network.

The three groups are :

- **Client**

This part contains the DB2 Client to connect to the data warehouse server, Design Studio to develop an end-to-end solution, query patroller Center to control the Query Patroller from a graphical interface, Intelligent Miner Visualizer to have a graphical representation of the data mining results and the Miningblox.

- **Data Warehouse server**

This part is the RDBMS side. There are also other installed products like the Intelligent Miner, the query patroller, the cube views.

- **Application server**

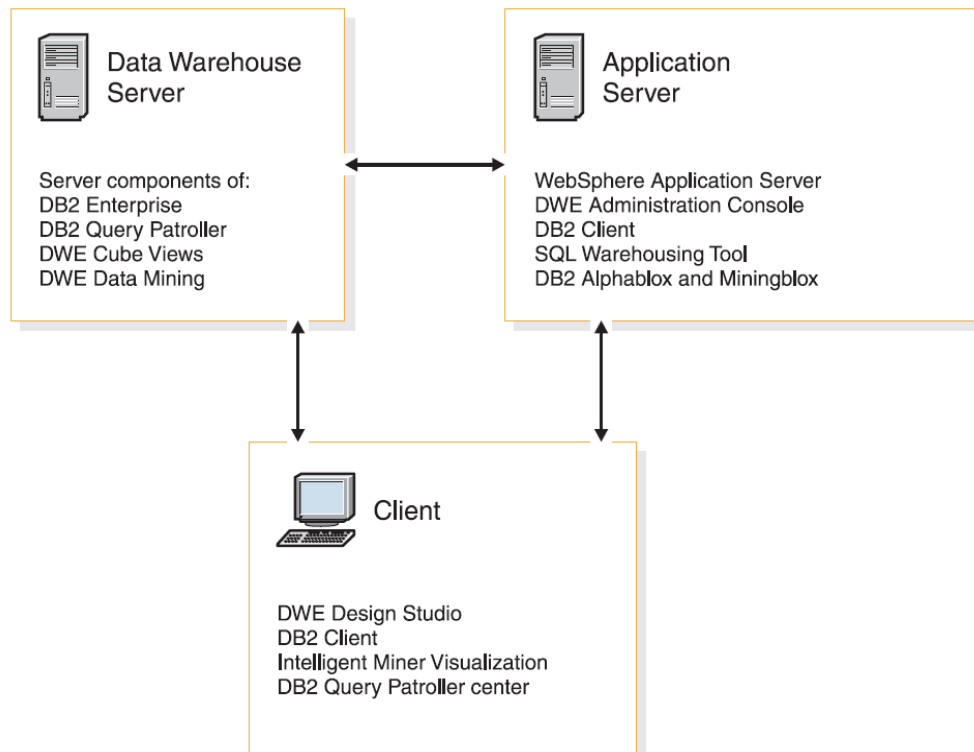


Illustration 2: DWE Architecture.

3.3 DB2 DWE components

The advantage of the I.B.M solution is that all components are integrated. Each component works with the others and everything is transparent for the developer.

Here is a brief description of some components and softwares, we will use all over the project:

- **DWE Design Studio**

Design Studio is a Integrated Development Environment which is based on the Eclipse IDE. We will use it to design physical data model. This designing process in Design Studio is based on the Rational Data Architect (Rational Rose).

Design Studio Integrates the SQL Warehousing tool which is used while designing data flows and control flows. It is possible to deploy control flow on the Websphere application server just with one click on a button.

The environment provides also tools for data mining, OLAP metadata and cube models.

- **DWE SQL Warehousing tool**

The SQL Warehousing tool provides a new way for the programmer to code application process.

In Design Studio, the developer will design graphics representing data transformations, data flows and control flows. SQL Warehousing tool can automatically generate SQL from the designed graphics.

- **DWE Administration console**

The DWE Administration console is an application that is deployed on Websphere. The console provides an administrating tool to manage the application on the server.

The administrator access to the console from his browser.

<http://appsvr:9080/dweadm/>

- **DB2 Enterprise Server Edition**

DB2 Enterprise Server Edition is a multi-users relational database management system that is specially tuned to manage data warehouse.

This version is specially tuned to manage data warehouse because of features like “data compression capabilities” that reduces the requirement of big disk space.

- **DB2 Alphablox**

DB2 Alphablox is a tool to create web based applications which report information from the database, from a cube view or from a data mining process.

4 THE BUSINESS SCENARIO

4.1 The fictional company

4.1.1 Presentation of the JK Superstore

The goal of my work is to explain how to build an end-to-end business intelligence solution. I.B.M provides a sample database. This sample is a fictional company called JK Superstore.

To make tutorials more interesting and things more realistic, we belong now to the JK Superstore developers team.

JK Superstore is a company that owns retail stores. They sell shoes, beauty, home furnishings, clothes and electronics.

4.1.2 Understanding of the need of the company

The company grows very fast and the director has plan to extend the business to new markets. But before doing it, he wants to make sure that it will be a benefit for the company and also that he will keep the same profitability during the expansion.

The director ask to his programmers team to make possible to analyze the current situation of the company and to make decisions thanks to these analyses.

4.2 The sample database

4.2.1 The current technical situation of the company

The company is using I.B.M Data Warehouse Professional Edition. The developers team has designed two databases. The first one is the data warehouse and the second one is the Marts Schema. The latter one will be used to make analyzes.

4.2.2 The Data Warehouse Schema

This schema contains transactional data of the different stores. In fact, it is a “Market Basket” model where the data of all stores are consolidated.

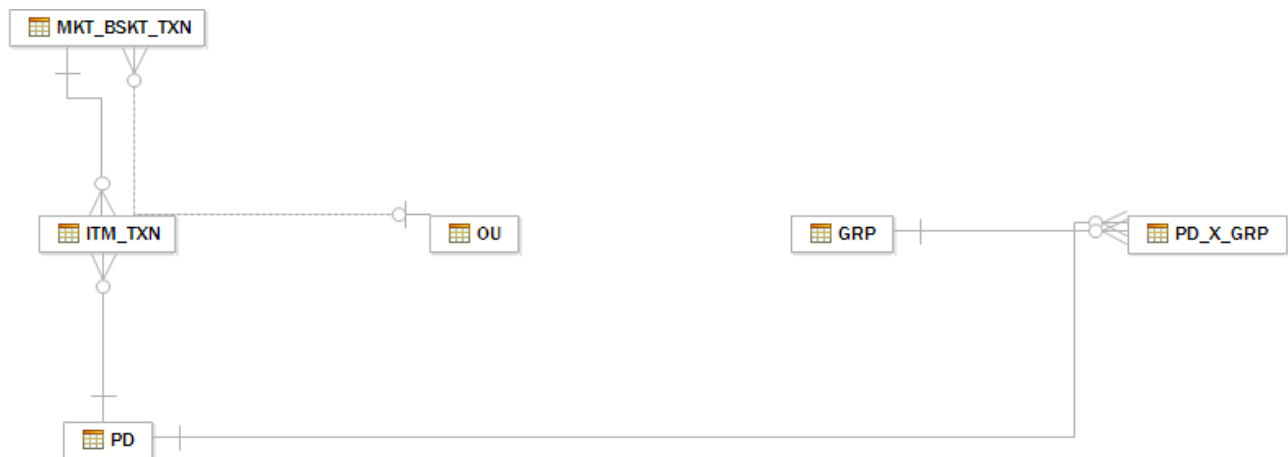


Illustration 3: The DWH Schema.

The complete schema contains nine tables. Here is a brief description of the tables and their roles in the schema:

The Item Transaction table (ITM_TXN) contains items of a transaction. An item of a transaction can be one bar code scanned at a checkout or a customer that want a refund for an item. This is one item(s) line on the receipt.

The Market basket transaction table (MKT_BSKT_TXN) regroups items bought by a customer at one moment into a single event. For Example, you have buy “Nescafe” and water at the DIA Market to make a “frappe”¹ . This is the bill.

The Organization Unit table (OU) is loaded with the different stores that belong to the JK Super Store. The table contains a hierarchy in five levels. The levels are organization level, sub-divisions level, regions level, districts level and in the end the stores level.

The Product table (PD) is designed to contain the different products provided by JK Super Store.

The Group table (GRP) contains the different groups of products. A group can be divided into groups.

The Products by Group table (PD_X_GRP) contains the relationships between products and groups.

The Involved Party table (IP) contains people which are related to the JK Super Store business activities (store managers, distributions contacts).

The Customers table (CL) records Id’s of the customers. They are anonymous but we will need their id to make data mining exercises.

The Measurement period (MSR_PRD) records the intervals of time at which measurements are captured in the warehouse.

Only the ITM_TXN table and the MKT_BSK_TXN table are empty. We will load them with data received from all the stores. The stores send to the servers what they sold and what they refund.

¹ Frappe is a Greek invention. It is cold coffee with cream.

4.2.3 The Data Marts Schema

4.2.3.1 What is a dimensional model

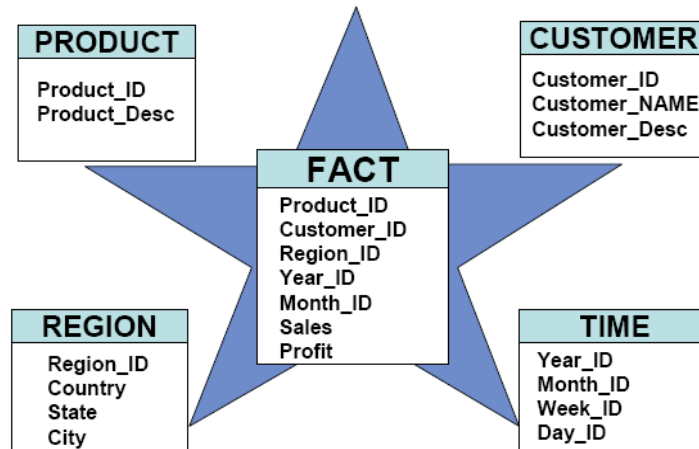


Illustration 4: Example of a star schema

This kind of model is also called star model and is the most famous model in data warehousing. The query performance are better with a star model than an E/R model and is easier to understand. It consists in 2 kind of tables.

The fact table is the central table of the star schema. The fact table contains the values the business want to analyzes. On the Illustration 4, the sales and the profits are the measurements. The other columns are references to dimensional tables.

The dimensional tables contain details about the facts. It enables us to have a better understanding of the data in the fact table.

It exists also other kind of dimensional schema (“Illustration 5”) like snowflake schema and multi-stars schema. In our cases, we are using mainly the star schema.

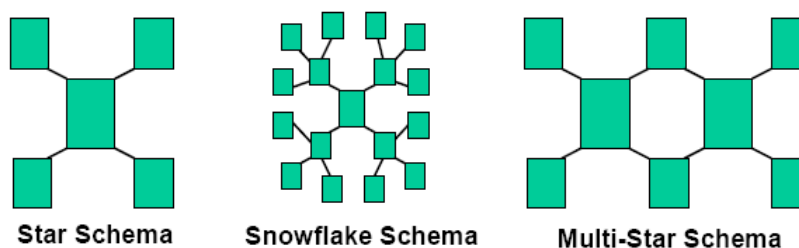


Illustration 5: Different types of dimensional models

4.2.3.2 The JK-Superstore Marts

This schema contains the aggregated data that will be used to make analyzes of the sales and prices of JK Superstore. The Marts schema is a multidimensional model and contains three dimensions and a fact table.

The three dimension tables are the TIME, STORE and PRODUCT and the fact table is called PRCHS_PRFL_ANALYSIS.

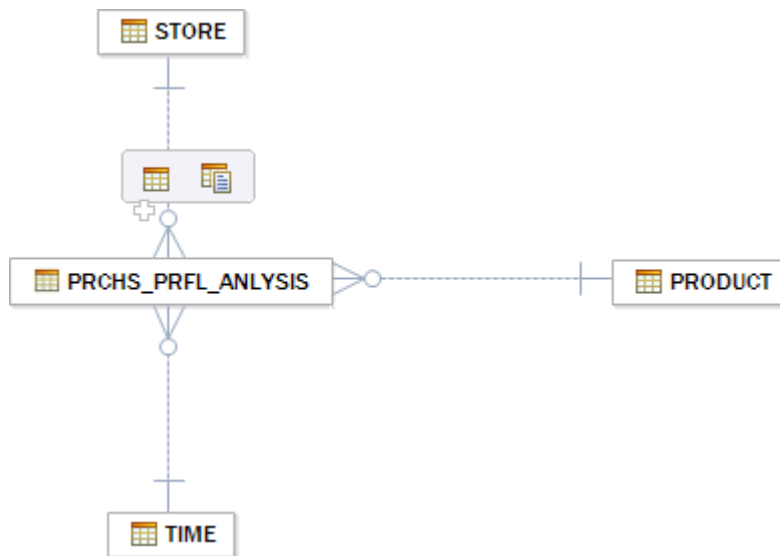


Illustration 6: The Mart schema

The purchase profile analysis fact table contains the following measurements:

Name of the column	Description
NUMBER_OF_ITEMS	A measure that identifies the count of individual product items.
NMBR_OF_MRKT_BSKTS	Number of customer visits.
PRDCT_BK_PRC_AMUNT	Price that the Organism Unit (Store) expects to sale the product.
CST_OF_GDS_SLD_CGS	Acquisition price.
SALES_AMOUNT	Value of the market basket.

4.3 Prerequisites before starting with tutorials

4.3.1 How to install DB2 DWE Edition 9.1.1

The installation is done from a graphical Java graphical interface and is friendly user but the prerequisites of the installation are not very well explained.

The installation files must follow a specific tree of directories. This tree is defined in the manual but is not up to date and is not working with the current version.

Here is the tree I had to create to start the installation:

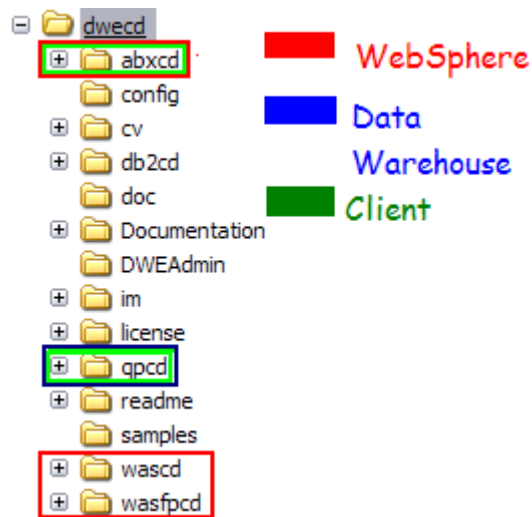


Illustration 7: Installation Tree

The coloration shows the parts that each component requires at the installation. The uncolored parts are needed parts no matter of the component we will install.

4.3.2 Creation of the database

To query the database, we use the DB2 Command Windows (It is like SQL Plus for Oracle). We will use it to create, to erase and to connect to a database. The DB2 Command Windows starts an Command Line Processor (CPL) that we can use in three different modes².

The three modes are :

- The interactive mode open a DB2 prompt ==>.
- Command Mode. From the DOS batch, we can execute a command by preceding it with DB2.
- Batch Mode. It is like Command Mode but the option -f import and execute the content of a file.

For the creation of the database, the Command Mode will be used. The syntax in Command Mode is the following:

Code: Syntax of the DB2 in Command Mode

```
db2 [command [database][option]]
```

The JK-Superstore database is called DWESAMP. So first we will create it. To avoid an error by trying to create a database that already exists we will try to delete the DWESAMP database first.

Code: Suppression, creation and connection to the JK-Superstore database

```
-- Suppression of the database
db2 drop db DWESAMP
-- Creation of the database
db2 create db DWESAMP
-- Connection to the database
db2 connect DWESAMP
```

² I.B.M. DB2 Command Window Documentation can be found at
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0010409.htm>

We have now to create the data warehouse and the Marts schema³. To do it, we will use the DB2 Command Windows in batch mode. In this mode, the db2 command executes the content of a file. In our case it will execute an SQL script that will create the tables, constraints and stored procedures.

Code: Syntax of the DB2 Command windows in batch mode.

```
DB2 -vf [filename]
```

The following code can be execute from a batch file:

Code: Part of the .bat file that create the DWE and MARTS schema

```
-- Creation of the tables of the DWE schema
db2 -tvf createwarehouseTables.sql
-- Load basic data in the DWH Schema from text files
db2 -tvf loadwarehouseTablesMinusItmTrns.sql
-- Add constraints on DWH schema
db2 -tvf addDWHri.sql
-- Add stored functions, that will be used by MARTS schema
db2 -tvf sqw\dwesamp_marts_func.sql
--Creation of the DWH Schema ( Table, Column and Comments)
db2 -tvf createMartTables.sql
-- Load MARTS tables from data files
db2 -tvf loadMartTablesTimeProduct.sql
--Count rows added and check if everything was well done.
db2 -tvf countRowsSQW.sql
```

If the execution of the batch file has been successful, then the execution of the last should return this result:

TABLERNAME	ROWS	COMMENTS
DWH_MSR_PRD	1303	OK ?
DWH_OU	26	OK ?
DWH_IP	69495	OK ?
DWH_CL	105	OK ?
DWH_PD	35259	OK ?
DWH_GRP	13514	OK ?
DWH_PD_X_GRP	35259	OK ?
DWH_MKT_BSKT_TXN	43165	OK ?
DWH_ITM_TXN	0	OK ?
MARTS_TIME	1288	OK ?
MARTS_STORE	0	OK ?
MARTS_PRODUCT	35259	OK ?
MARTS_PRCHS_PREFL_ANALYSIS	0	OK ?

³ It is possible to add comments in the database about object thanks to COMMENT ON object IS.

Example : Here is a comment on a column :

```
COMMENT ON COLUMN "PD_ID" IS 'The unique identifier of the Product';
```

5 DESIGN STUDIO TUTORIALS

5.1 The Eclipse based Integrated Development Environment

5.1.1 Introduction to the Design

Design Studio is the Integrated development Environment that I have used all over my project. The provided environment is clear and allows us to do a lot of tasks on the Database and on the Websphere application Server. Design Studio is based on the Eclipse IDE but it includes more functionalities.

5.1.2 Projects in Design Studio

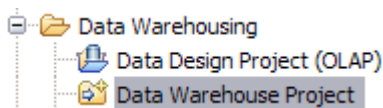
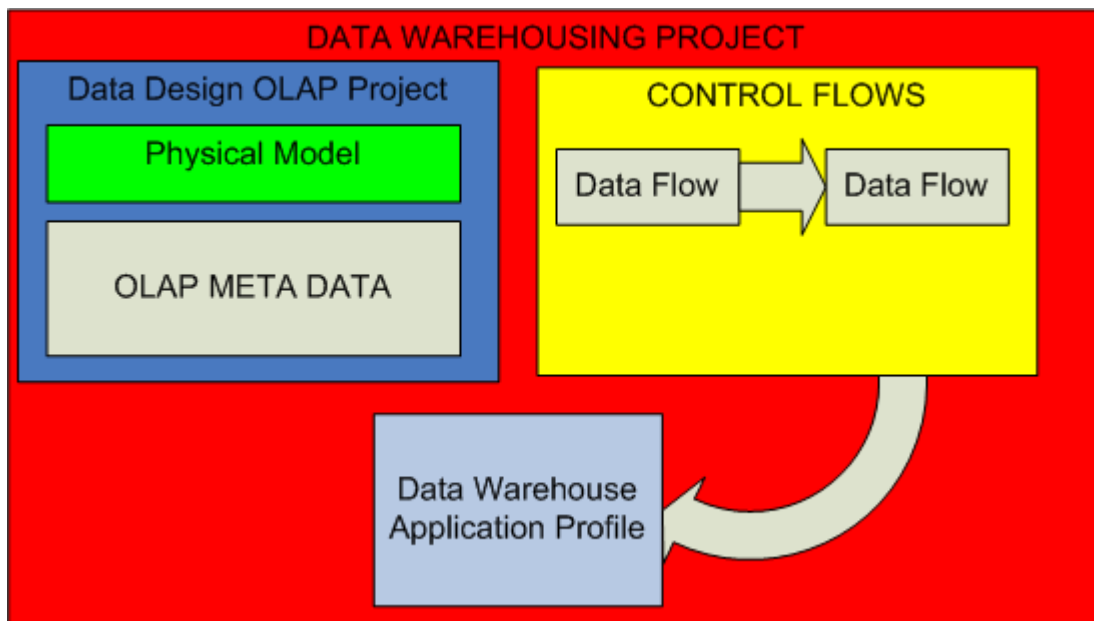


Illustration 8: Data Warehousing projects

If we compare Design Studio to Eclipse, the difference is in the projects we can create. Design Studio have two kinds of projects Eclipse do not have (see “Illustration 8”). These projects are data warehouse oriented project. The data Design Project have now the capabilities to add OLAP meta data in the database.

We will create in Design Studio different projects so that we can work on different part of our business intelligence solution. For example, we will create a Data Design Project (OLAP) to work on the physical data model and then create a data warehousing application that will prepare the database for data mining.



5.2 Global aim of these tutorials

In the following pages, we will start tutorials. Our goal will be to develop an data warehousing application and to do data mining on the JK-Superstore data so that the director can make decisions and have a better understanding of how to improve his benefits. We will start from the scratch with an almost empty database.

5.3 The tutorials and their goals

First, we will create a Data Design Project. In this project, we will work on the physical data model by adding constraints, comparing the schema from the database with the one in Design Studio, validating of the schema and propagating changes on the database.

After that, we will create a Data Warehouse Project. With it, we will design data flows, control flows to Extract Transform and Load data in the database. We will also deploy the control flow on the application server. This is the data warehousing application. The aim of the data warehousing application is to build the data warehouse and feed it with the daily data of the stores. And Finally, we will design Data Mining Flow.

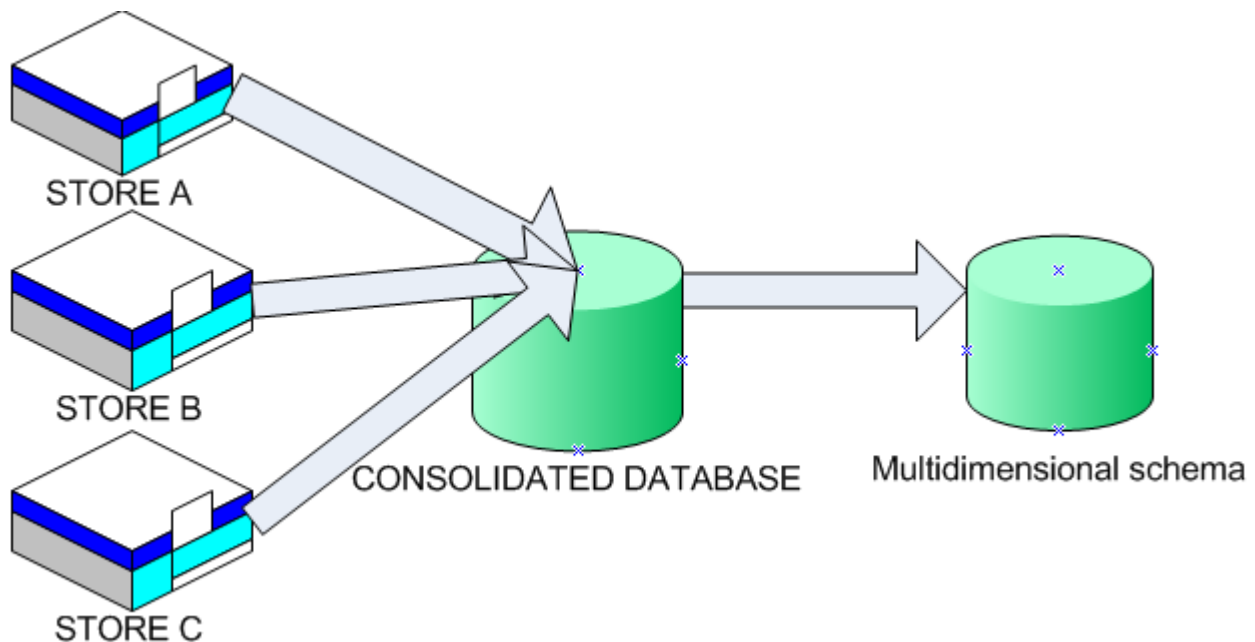


Illustration 9: Flow of the data from the stores to the MARTS schemas.

The stores in the tutorial are materialized by files. If we were in a non fictive scenario, the file import operator, we use would be changed to a table source.

6 THE PHYSICAL DATA MODEL

6.1 Introduction and prerequisites

In this tutorial, we create a data design project that contains a data model. This data model will be a copy of the DWH and MARTS schema we created previously. The copy process is called Reverse Engineering. After that, we will add constraints on tables of the MARTS schema, validate the modified schema and propagate the modifications on the database.

Prerequisites are first to be connected to the DWESAMP database, otherwise it is impossible to copy the database. The connection to the database is available from the database explorer panel (see Illustration 10). Right-click on the DWESAMP database in the connection folder and then click the button “Reconnect”. You will have to insert your User Id and Password to connect.

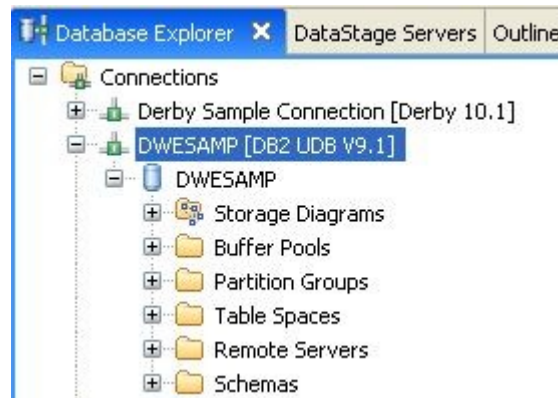


Illustration 10: Database Explorer View

Known problem: “I can't see my database in the Connections Folder”

Solution: You have to create a new connection to you DB2 Server. This may arrive if the server is not running on the same computer. You have to create a new connection only once.

6.2 Creation of the data design project

To create a data design project, we have to click on File -> New Project and open the Data Warehousing Project directory. We Choose the Data Design Project(OLAP) and name it “Data Model”.

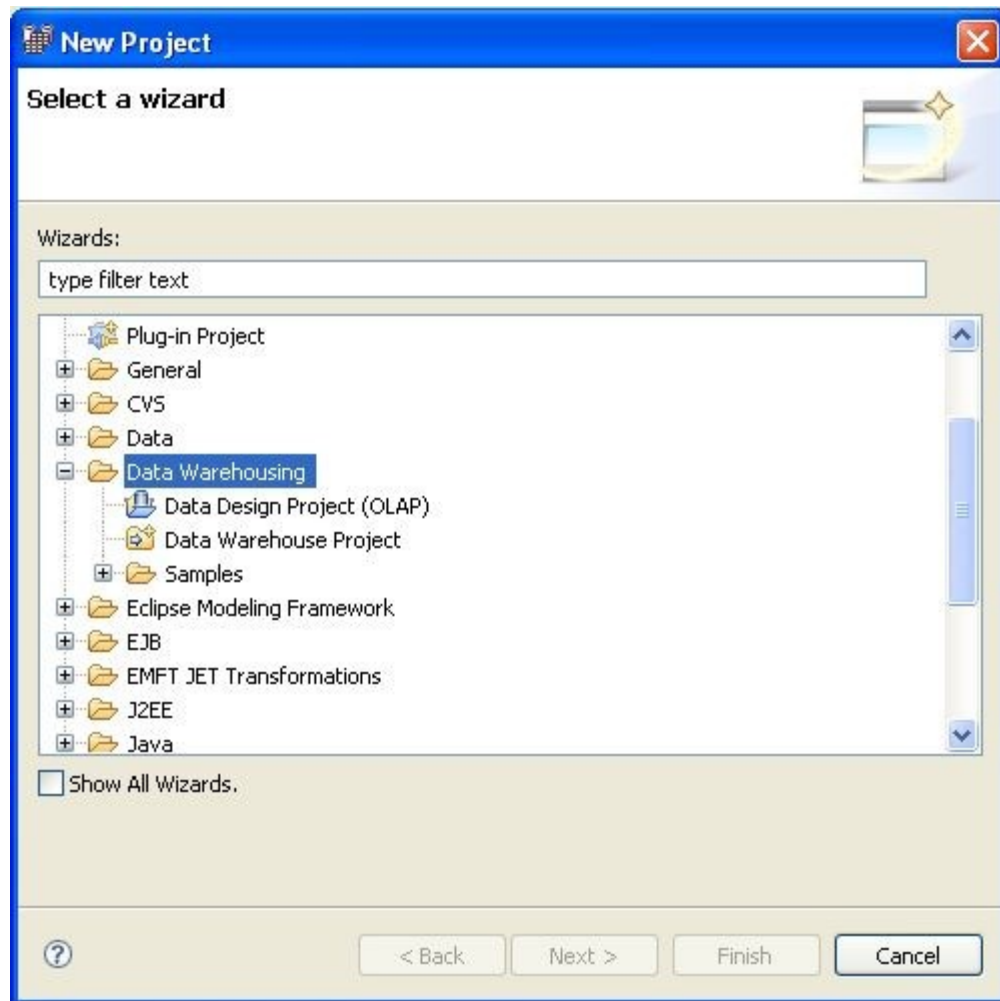


Illustration 11: New Project Wizard Window

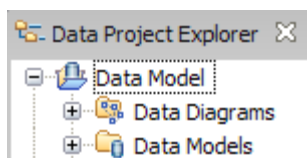


Illustration 12: Data Design Project in the Data Project Explorer

When the data design project is created, it is shown in the Project Explorer view.

6.3 Reverse Engineering the existing database

Now that the data design project is created and that the connection to the database is established, we can copy the model from the database into the project. In the Data Model project, we right-click the data models folder in order to create a new physical model (see Illustration 13).

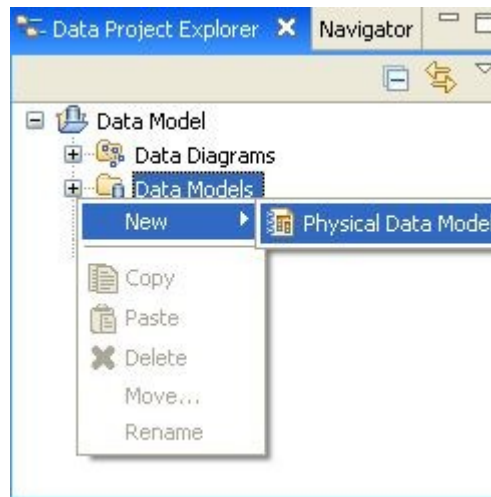


Illustration 13: Creation of a physical data model

A wizard will help us during all the process. In the first window, check the reverse engineering radio button and set the database to DB2 UDB (Universal Data Base) and version 9.1 (see Illustration 14).

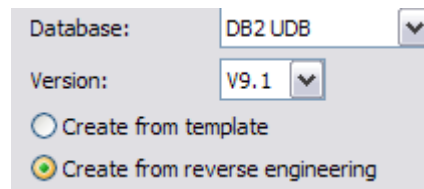
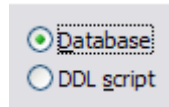


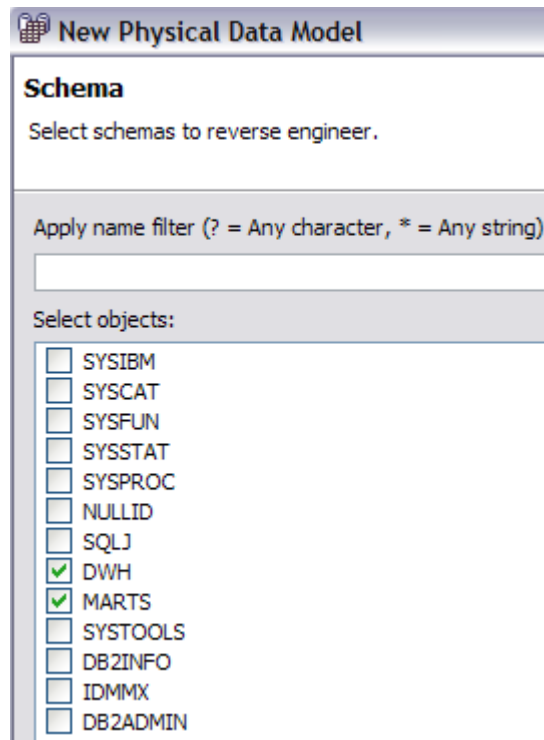
Illustration 14: First page of the wizard

On the next screen, we can choose the source of the copy. We can create a data model from a database but also from a DDL (Data definition Language) script. A DDL Script is, for example, the scripts we have executed before from the Command Window in order to create the database. This script contains CREATE statement that can be reverse engineering to a data model in Design Studio. But in our case, we will use the previously established connection to copy it from the database.

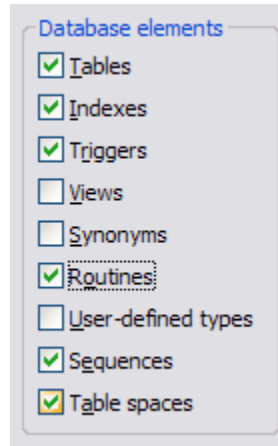


On the next screen, select the existing and established connection to the DWESAMP database.

The wizard will now connect to the database and retrieve the list of schema. On this screen, the selected schema corresponds to the schema that will be reverse engineered. We are only interested in DWH and MARTS schema.



On the next Screen, we will have to choose the types objects that we want to copy:



The reverse engineering operation can take a while. When it is done, we can extend the “Data Models” directory and notice that our physical data model has been imported.

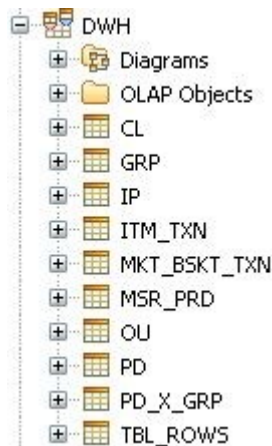


Illustration 15: Data Warehouse schema.

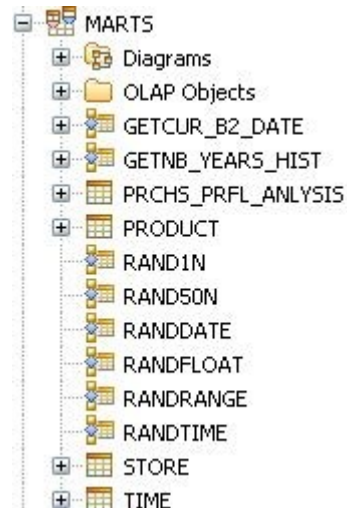


Illustration 16: MARTS schema.

It is possible to have a graphical representation of both schemas from the Data Diagrams folder in the project explorer.

Known problem: “The Data Diagram directory is empty.”

Solution: This problem can occur if the Data Model is not extended. By Clicking on the [+] in front of the data model from the data models directory in the project, you can resolve the problem and then go back

again in the data diagram and see the DWH and MARTS diagram. (The Data Model must be opened and extended to have diagrams in the data diagram directory).

6.4 Working on the physical data model

6.4.1 Defining the problems

A closer look the MARTS schema let us notice that there is no primary keys and no associations between tables in this schema. The schema is not normalized.

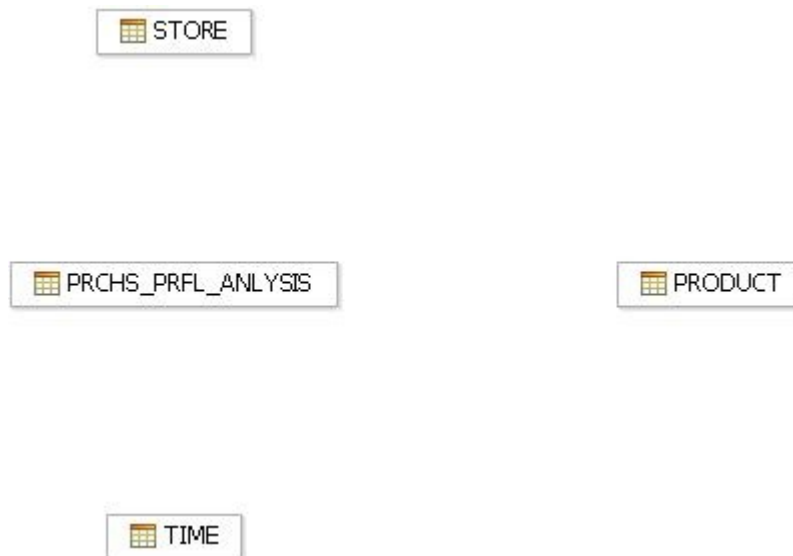


Illustration 17: The MARTS Data Diagram shows that there is no associations between tables

We will first add the primary keys on the dimensional tables, then the foreign key constraints.

6.4.2 Adding Primary keys

The STORE, PRODUCT and TIME table have no primary key. To resolve this problem, we will add foreign from the Column tab in the Properties view of each tables.

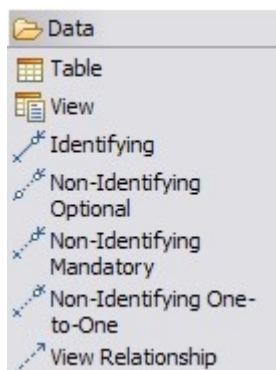
Name	Primary Key	Data Type	Length	Scale	Not Null	Generated	Default Value/Generate E...
STR_IP_ID	<input checked="" type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input type="checkbox"/>	
STR_TP_NM	<input type="checkbox"/>	VARCHAR	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
ORG_IP_ID	<input type="checkbox"/>	INTEGER			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
PRN_OU_IP_ID	<input type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input type="checkbox"/>	
MGR_EMPE_ID	<input type="checkbox"/>	INTEGER			<input type="checkbox"/>	<input type="checkbox"/>	
NR_CPTR_PRX_NM	<input type="checkbox"/>	VARCHAR	64		<input type="checkbox"/>	<input type="checkbox"/>	
SALE_VOL_RNG_...	<input type="checkbox"/>	VARCHAR	64		<input type="checkbox"/>	<input type="checkbox"/>	
FLRSP_AREA_RN...	<input type="checkbox"/>	VARCHAR	64		<input type="checkbox"/>	<input type="checkbox"/>	
STR_CODE	<input type="checkbox"/>	CHAR	6		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
STR_SUB_DIV_NM	<input type="checkbox"/>	VARCHAR	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
STR_REG_NM	<input type="checkbox"/>	VARCHAR	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
STR_DIS_NM	<input type="checkbox"/>	VARCHAR	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
STR_NM	<input type="checkbox"/>	VARCHAR	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Illustration 18: Column property view of the STORE table

The primary key constraint is added by checking the primary button for the correct column. For example, the STORE primary key column is STR_IP_ID.

6.4.3 Adding Associations - Foreign keys

The associations between tables can be done following different methods. The Data Diagrams palette provides tools to create all the needed types of associations between tables. It is possible to create identifying associations, and non-identifying associations optional, mandatory and one-to-one.



Another way to create associations is to use the “Show Implicit relationships” option. This option is accessible from the Data Diagram by right-clicking on a white part of this one. The implicit relationships option uses the name of the columns to create associations. This can so only be used if the name of the columns are similar in the fact table and in the dimensional table.

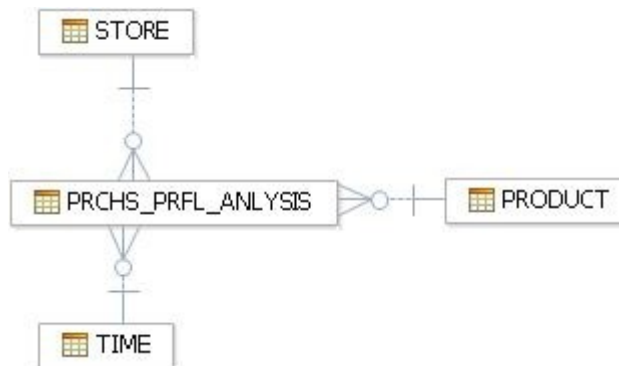
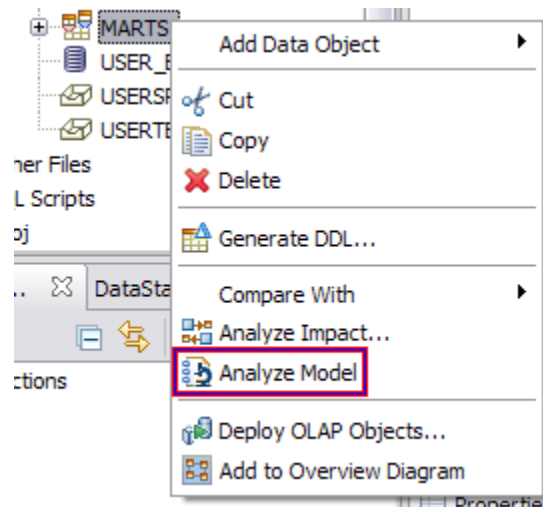


Illustration 19: Completed MARTS diagram

6.5 Validation of the model

Before propagating the modifications to the database and maybe cause problems, a validation of the model can be done to avoid error to occur on the database. A problem is less important in Design Studio than on the database where we could cause problems to database users and maybe also customers.

The validation process is done by right-clicking the MARTS schema in the Data Model.



The result of the analyze model process can be seen in the Data Output view. Hopefully we ran an analyze of the model because there were errors. The name of the automatically generated foreign key constraints are too long. The maximum length of object names in DB2 is limited at 18 characters. We will have to change the name of the constraints with shorter ones and then run again a analyze of the model.

Errors (3 items)			
✘	The name length of PRCHS_PRFL_ANALYSIS_PRODUCT_FK exceeds identifier length limit which is 18	DWESample...	PRCHS_P...
✘	The name length of PRCHS_PRFL_ANALYSIS_STORE_FK exceeds identifier length limit which is 18	DWESample...	PRCHS_P...
✘	The name length of PRCHS_PRFL_ANALYSIS_TIME_FK exceeds identifier length limit which is 18	DWESample...	PRCHS_P...

The validation of the model is not an obligation in Design Studio, but it is better to resolve the problem at the source than on the database where other persons can be disturbed.

6.6 Propagation of the modification on the database

6.6.1 The Comparison process

Now that the model is cleaned, it will be compared with the current database model. The comparison is useful because it provides an overview of the modifications that have been done. This is a kind of report before deploying the changes on the database.

The comparison option can be found by right-clicking the MARTS schema and choosing “Compare with original source”. The reports look like the Illustration 20.

Item	Data Model/DWESample.dbm/DWESAMP.MA...	DWESAMP:DWESAMP.MARTS
Schema	<Schema> MARTS	<Schema> MARTS
Table	<Table> PRCHS_PRFL_ANALYSIS	<Table> PRCHS_PRFL_ANALYSIS
Foreign Key	<Foreign Key> FACT_PRODUCT_FK	
Foreign Key	<Foreign Key> FACT_STORE_FK	
Foreign Key	<Foreign Key> FACT_TIME_FK	
Table	<Table> PRODUCT	<Table> PRODUCT
referencingForeign...	name:FACT_PRODUCT_FK,description:null,label:n...	
Primary Key	<Primary Key> PRODUCT_PK	
Table	<Table> STORE	<Table> STORE
referencingForeign...	name:FACT_STORE_FK,description:null,label:null,...	
Primary Key	<Primary Key> STORE_PK	
Table	<Table> TIME	<Table> TIME
referencingForeign...	name:FACT_TIME_FK,description:null,label:null,de...	
Primary Key	<Primary Key> TIME_PK	

Illustration 20: Comparison of the Models / The one on the right is the current situation on the database.


This report shows there were no foreign and primary keys on the DWESAMP:DWESAMP.MARTS schema but they exist on the Data Model. From this moment, the modification can be done in two directions. We can discard the modifications in the data model (it will change the schema so that it is the same as on the database), or propagate the changes on the database. In Design Studio these two options are called from right to left or from left to right.

Our goal is to propagate changes on the database, so the right to left operation button is pressed . On the report the modifications have now been propagated but they are still not in the database.

Note: The From left to right button will only work with the selected changes. In our case we have to first select the Schema item then click the button. This is because it is possible to only propagate some modifications and not all.

6.6.2 *The DDL script*

In the comparison process, we choose the changes that will be propagated. Design Studio updates the schema on the database thanks to a DDL script that contains ALTER statements. The advantage is that the generated script is a differential script that will update the database and not remove everything and then create all the objects from the scratch.

In order to generate the script, we click on the “Generate right delta DDL” button  and check the “Run DDL on the server” radio button. The generated script is run on the database and will be saved in the SQL Scripts directory of the project.

Code: The DDL scripts generated after the previous modifications

```
ALTER TABLE "MARTS"."PRODUCT" ADD CONSTRAINT "PRODUCT_PK" PRIMARY KEY ("PD_ID");

ALTER TABLE "MARTS"."STORE" ADD CONSTRAINT "STORE_PK" PRIMARY KEY ("STR_IP_ID");

ALTER TABLE "MARTS"."PRCHS_PRFL_ANALYSIS" ADD CONSTRAINT "FACT_STORE_FK" FOREIGN
KEY ("STR_IP_ID")
    REFERENCES "MARTS"."STORE" ("STR_IP_ID");

ALTER TABLE "MARTS"."PRCHS_PRFL_ANALYSIS" ADD CONSTRAINT "FACT_TIME_FK" FOREIGN KEY
("TIME_ID")
    REFERENCES "MARTS"."TIME" ("TIME_ID");

ALTER TABLE "MARTS"."PRCHS_PRFL_ANALYSIS" ADD CONSTRAINT "FACT_PRODUCT_FK" FOREIGN
KEY ("PD_ID")
    REFERENCES "MARTS"."PRODUCT" ("PD_ID");
```

7 THE DATA FLOWS

7.1 The data warehouse project

7.1.1 Introduction

The data warehouse project is the second type of projects that Design Studio adds to the Eclipse IDE. The data warehouse project provides tools to create data warehousing applications. A data warehousing application is an application that is deployed on the Websphere application server. This application runs control flows that contains and orders data flows. Data warehousing applications are used to maintain the database up to date with the new data and to automatize tasks.

The work on the data design project is now temporally finished , we will come again with this project in order to create OLAP cube model, OLAP cubes and MQT's.

7.1.2 What provides this project?

This project provides a lot of features that we will used to create a data warehousing application. Data flows, control flows and mining flows are designed in this project. The Data Design project, we created previously, is used as a referenced project in the data warehouse project because if a data flow loads data in the database, it needs to know how are the tables on the database. The Data Design project is a part of the the data warehouse project and this referenced is defined in the Database directory in the DW project.

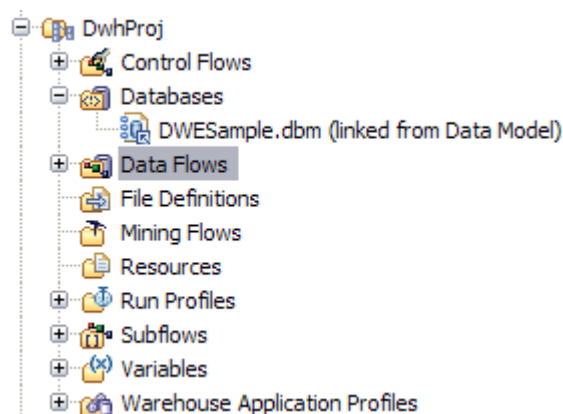


Illustration 21: A data warehouse project linked with a Data Design project (See Databases directory) .

7.2 Definition of a data flow in Design Studio

A data flow is a graphical model that is used to design SQL-based data extractions and transformations. It is created by moving and connecting data flow operators on the canvas. Almost all operators have inputs and outputs ports that stand for the connections between operators.

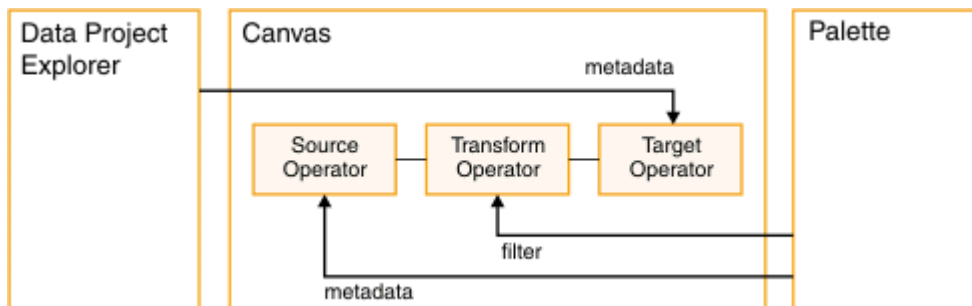


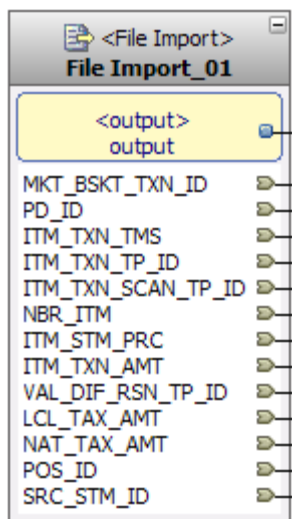
Illustration 22: A basic data flow in the Design Studio data flow editor. Metadata from a linked data design project is used to define the source and target tables. Operators can be dragged and dropped from the palette. (Image and caption from I.B.M infocenter)

The SQL Warehousing Tool can transform a data flow in SQL queries. This is the based structure of the warehousing application development (see further for more information about control flows and data warehousing applications).

7.3 Definition of some data flow operators

7.3.1 File Import operator

The file import operator is used to extract data sets from a flat file that must be formatted following specific rules. These rules must be defined in a special file that has the extension “.fileformat” that contains information about the structure used in the flat file (This fileformat is in fact a XML Schema).



The File Import Operator has only one output port that other operators can connect to.

In this example, the used fileformat is based on the MKT_BSKT_TXN table.

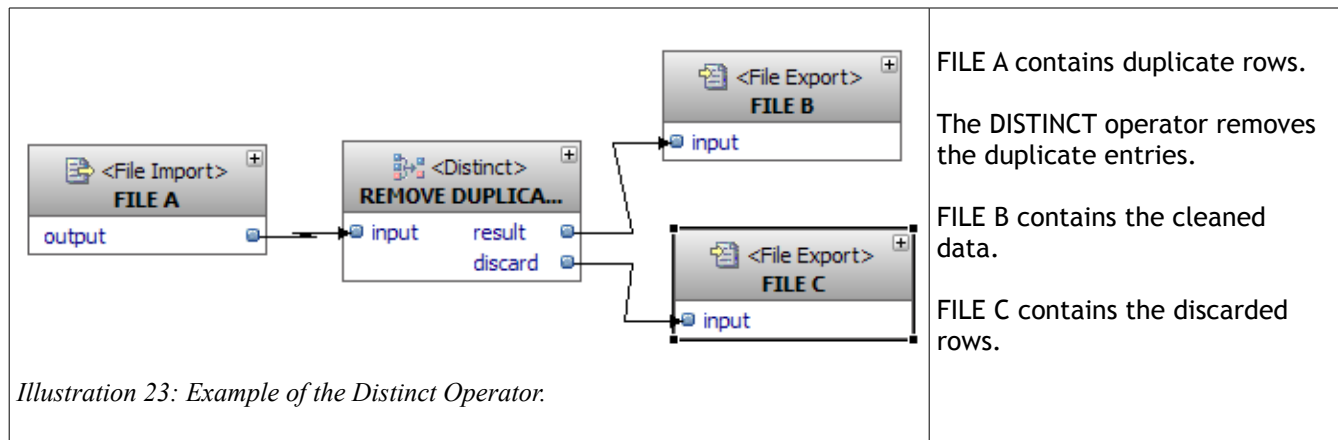
7.3.2 File Export operator

This operator does the opposite of the previous one. It exports data sets into delimited flat files. In the next tutorials, we are going to use this operator in order to hold the discarded data by other operators. For example, a Distinct Operator may remove rows from a data set. The removed rows will be retrievable from the discard output that we will be connected to the input port of the file export operator.

7.3.3 Distinct operator

The distinct operator removes duplicate rows from a data set. In the properties view, the “Column Select” tab allows to choose the column that will be used for the distinction.

Example: Removing duplicate entries in a file.



7.3.4 Union operator

The union operator combines two data sets sharing the same structure into one data set. The union operator can do the following set operations: Union, Intersection and Exception⁴. (ambiguity with the operator name)

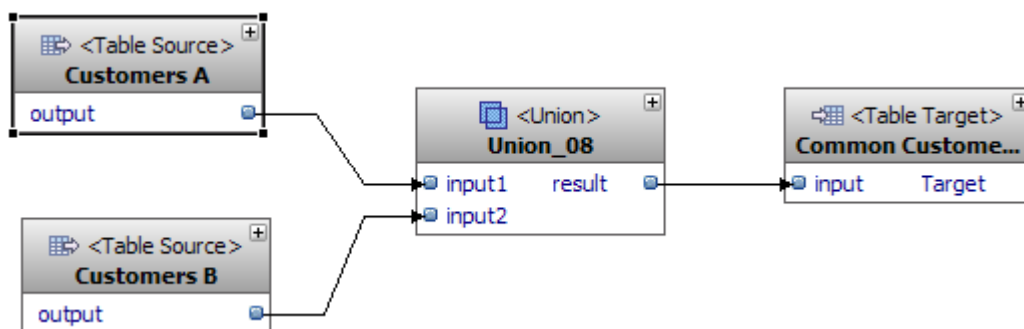


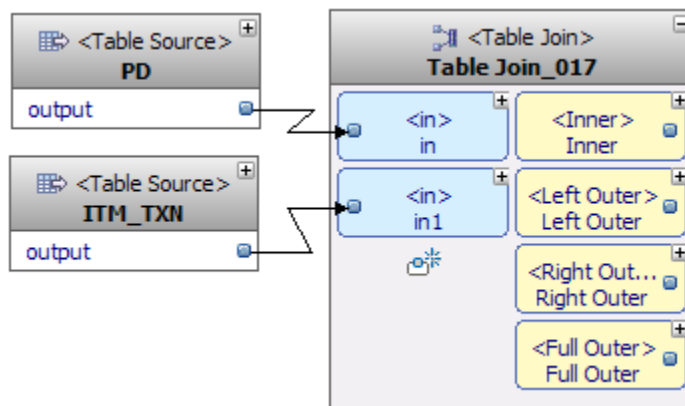
Illustration 24: Abstract example to get customers that are registered in store A and B.

⁴ For more information about the set theory : http://en.wikipedia.org/wiki/Set_theory, http://en.wikipedia.org/wiki/Intersection_%28set_theory%29, http://en.wikipedia.org/wiki/Union_%28set_theory%29 and http://en.wikipedia.org/wiki/Complement_%28set_theory%29

7.3.5 Table Join operator

The table join operators do a Cartesian product between input tables. The Join condition is defined in the Condition tab of the property view.

Example: Simple Join between the product table and the Item Transaction Table.



The product table is joined with the ITM_TXN table.

The Condition is build with the SQL Condition Builder (see further) and is defined to :

```
IN_017.PD_ID = IN1_017.PD_ID
```

The Table Join operator has also a Select List option that allows to select the column in the projection part of the query.

Illustration 25: Example of the Table Join operator:

An advantage of this table join operator is that different operators can connect to different outputs. This is improving the performance because the Cartesian product is done only one time.

7.3.6 Select List operator and Where Condition operator

The select list operator removes or modifies column from a data set. It is the same as choosing the different column present in the SELECT of an SQL Query⁵. This operation is called a projection in SQL.

The Where Condition operator removes rows of a data set. This operation in SQL is called the SELECTION and must not be confused with the projection that uses the SELECT in the query.

⁵ For more information about projection and selection : http://nte-socio.univ-lyon2.fr/Marc_Grange/SQL1_en.htm

7.3.7 Key Lookup operator

The key lookup operator compares values of keys from the input table with the value of primary keys in one or more lookup tables and removes from the input table rows that do not contain matching key(s) in the lookup tables.

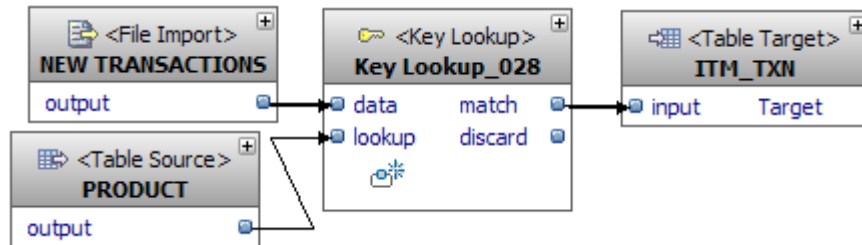


Illustration 26: Abstract example of the Key Lookup Operator.

In the Illustration 26, the file import operator feeds the Item Transaction table with the sales of the day. The key lookup operator will check if the sold product exists in the PRODUCT table.

7.3.8 Easy to understand operators

Some operators does not need any deep explanations because only their names are enough to understand what they do. Here is a non-exhaustive lists of other operators with a brief description:

- Order by operator

Order rows in data set depending on the values of one ore more column. The sorting can be ascending or descending.

- Group by operators
- Table Source and Table target operators

The table source operator creates a data set with the data from the table. In the other hand, the table target operator feed a table with the data in the input port.

- Splitter: The splitter splits a data set into two data sets. The operator property view have a Condition tab that is used to rules the separation of the data set.
- Table Target Operator and the Bulk Load Target Operator

Both Operators are used to load data in tables. The difference between the operators is the way they insert data in the table. Each operator provides different properties.

7.4 Data flows in the JK Superstore

Every day, all JK_Superstore markets sell and refund products to customers. The stores records the transactions in their own database and every night at 12'00 the data are sent to the data warehouse server that contains the consolidated database.

A first data flow is required to feed the Item Transaction table and the market basket transaction table with the daily sales of JK Superstore markets. This data flow gets data from flat files (If it was a database, we just have to replace the file import operator by a table source operator), transforms the data so that they are cleaned for the database and then load data in the tables.

When the data are consolidated in the DWH schema, we can feed the MARTS schema with the new data. This will be done with others data flows. There are three data flows, one for each table in the MARTS schema. In this tutorial, the fact table and one dimensional table will be loaded.(When we can do it for one dimensional table, we can do it for the other. The goad is to understand the mechanism).

At this moment, we can already feel that the execution order of data flows is important. Loading fact table before dimensional table may cause problems. This problematic will be resolve in the next chapter called Control Flows.

7.5 Populating the Item Transaction table

7.5.1 Origin of the data and start point of the flow

In this example, the data are separated in two files. Both files are formatted following the same “.fileformat” file. In a non-fictive context, the origin of the data could be databases of stores. In this case, the data flow changes are easy and fast to modify because only the file import operator must be changed to a table source operator.

Sample: A line of a flat file that correspond to a transaction item.

```
41007,603,"2004-01-17-
16.21.40.000000",69,93,1,+000000000004.44,+000000000004.44,,,,,9999,,
```

Sample: A part of the “.fileformat” file that defines the content of the first column of the structure. It corresponds to the 41007 value in the previous sample.

```
<columns
xsi:type="com.ibm.datatools.etl.dataflow.baselib.srctgt:FileFieldDescriptor"
name="MKT_BSKT_TXN_ID" virtualType="INTEGER">
  <containedType xsi:type="SQLDataTypes:IntegerDataType" name="INTEGER"
primitiveType="INTEGER"/>
</columns>
```

When the data flow is created, two “file import” operators are moved on the canvas. In the properties view of each operator, the file name label must be changed to target the flat files. We have also to define the fileformat, we are going to use. If everything goes well, the field list will be filled (see Illustration 27 on the next page).

General

File Format

Column Select

Advanced Options

Partition Options

File Import_01

Define the file format

File type: Field delimiter:

String delimiter: Date format:

Time format: Timestamp format:

Code page:

Additional file type modifiers:

Field list:

Column Name	Data Type	Length	Scale
MKT_BSKT_TXN_ID	INTEGER		
PD_ID	INTEGER		
ITM_TXN_TMS	TIMESTAMP		
ITM_TXN_TP_ID	SMALLINT		
ITM_TXN_SCAN_TP_ID	SMALLINT		
NBR_ITM	SMALLINT		
ITM_STM_PRC	DECIMAL	14	2
ITM_TXN_AMT	DECIMAL	14	2
VAL_DIF_RSN_TP_ID	SMALLINT		
LCL_TAX_AMT	DECIMAL	14	2
NAT_TAX_AMT	DECIMAL	14	2
POS_ID	INTEGER		
SRC_STM_ID	SMALLINT		

Illustration 27: Defining a fileformat for a file import warehousing operator.

7.5.2 Cleaning the data

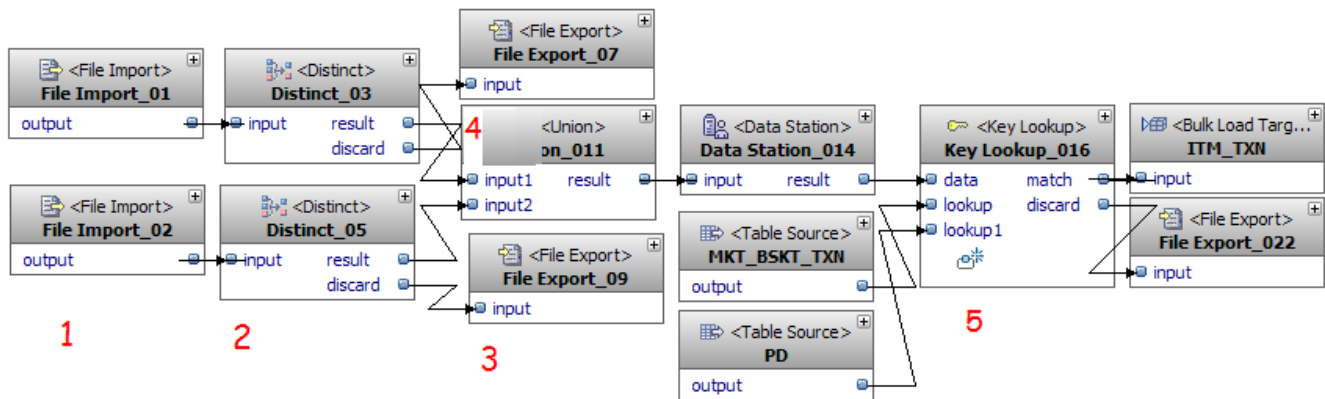


Illustration 28: Data flow that loads the ITM_TXN table.

Now that the data are on the flow **1**, we can start transforming and loading. First of all, in a flat file there is no verification of the duplicate data, it may contain twice the same entry. To avoid this problem (and to play with data flow operators), distinct operators are plugged **2** next to each file import operator. The discarded rows will be hold in files thanks to export file operators **3** (Same than for import file operator, the fileformat can be defined).

Next to the Distinct operators, both data sets are merged together in one data set with the union operator **4**. This unique data flow is now almost ready to be loaded in the ITM_TXN table but we have to check if the values in the foreign key column of the data set exist in the referenced table. For example, we check if the sold products exist in the PD (product) table **5**. This is done by using the Key look-up operator.

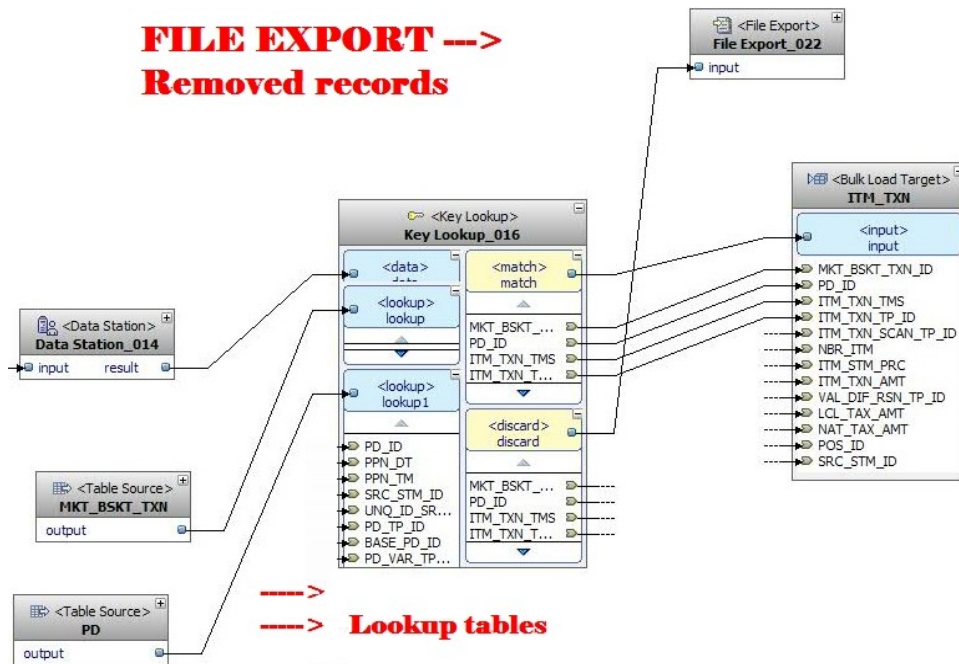


Illustration 29: Key lookup operators check the existence of foreign key(s) in lookup tables to prevent from enforcing the referential integrity.

For each lookup port of the key lookup operator, the matching condition must be build so that the operator know witch column must be compared. The condition is build with the SQL Condition Builder tool that simplifies things. This tool allows to create SQL Condition with the mouse by clicking on inputs, keywords, operations and functions.

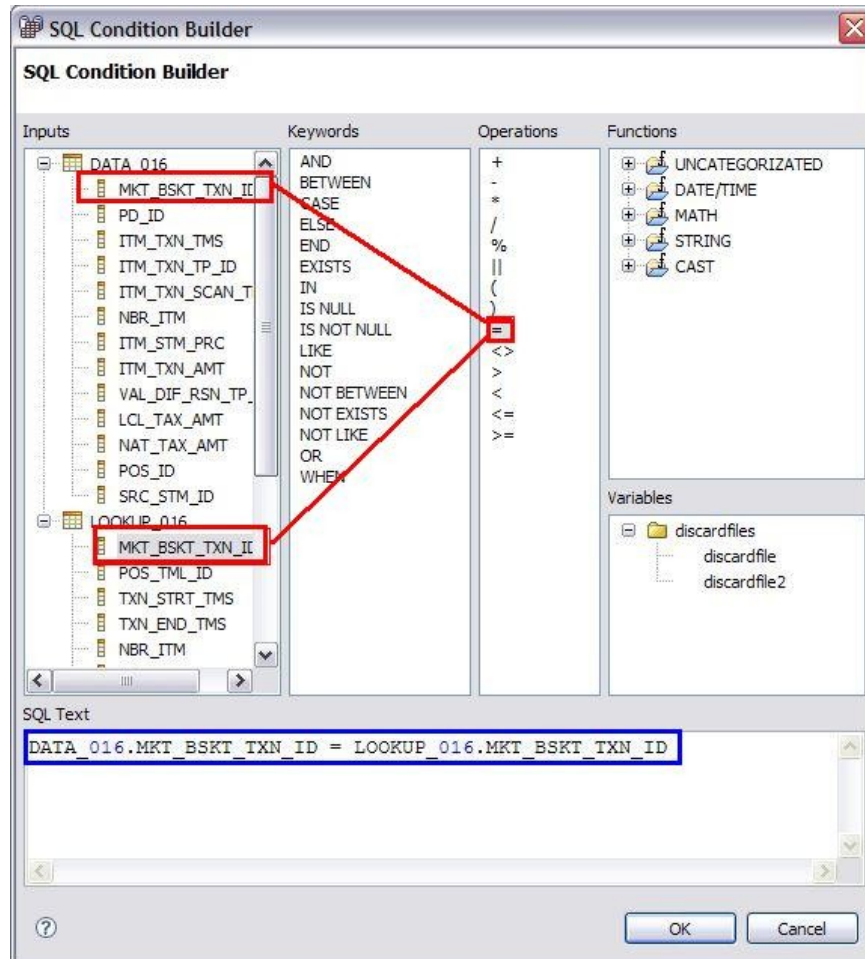


Illustration 30: SQL Condition builder.

The data station operator is an intermediate stage, where data can be filled into a staging table to check the flow content at any moment in the data flow. By right-clicking on the data station, we can retrieve data on the data flow. The content of the staging table is deleted before each run of the data flow.

7.6 Populating a dimensional table of the marts schema

7.6.1 Origin of the data

The dimensional tables from the MARTS schema will be loaded with transformed data from the DWH schema that contains consolidated data from the twenty-three stores of JK Superstore. In this example, we load the STORE dimensional table with the data from the Organization Unit table.

7.6.2 Explanation of the problem

The problem is that the Organization Unit table does not have only stores. The table contains a hierarchy of four levels.

Organization Unit Hierarchy

An organization contains sub-divisions
 Each sub-division is composed of regions
 Each regions owns multiple districts
 Each districts have one or more stores

The Hierarchy is done thanks to column of the table that contains the parent organization. When this column is null, it means that we are at the top of the hierarchy.

From the Organization Unit to the Store Table (Abstract Example)

<u>Example of data from the OU</u>	<u>How must be the data in the STORE table</u>
1 - Dia's Market - Parent: 2	1 - Dia's Market - North District - Gold Region - SubDivison -GoldOrga
2 - North District - Parent: 3	2 - Champion - North District - Gold Region - SubDivison -GoldOrga
3 - Gold Region - Parent: 4	
4 - SubDivision 1 - Parent: 5	
5 - GoldOrga - Parent: NULL	
6 - Champion - Parent : 2	

7.6.3 Resolving it within an SQL query

Before designing this complex data flow, here is the SQL query that corresponds to the transformations we do on the data.

Code: SQL query to transform data from organization unit.

```
SELECT STORE.OU_IP_ID, -- Id of the store
STORE.OU_CODE, -- Code of the store
STORE.ORG_IP_ID, -- Id of the organization the store belongs to
STORE.PRN_OU_IP_ID, -- Id of the Subdivision of the store
CUSTOMERS.NM, -- Id of the customers of the store
SUBDIV.OU_IP_ID, -- Id of the subdivision of the store -> In the data flow
replaced by the name (JOIN WITH IP)
REGION.OU_IP_ID, -- Id of the region of the store --> same as above
DISTRICT.OU_IP_ID -- Id of the District of the store --> same as above
FROM DWH.OU SUBDIV,DWH.OU REGION,DWH.OU DISTRICT,DWH.OU STORE,DWH.CL CUSTOMERS
WHERE SUBDIV.PRN_OU_IP_ID IS NULL --> LEVEL 1 Subdivision are rows without parent
AND SUBDIV.OU_IP_ID = REGION.PRN_OU_IP_ID --> LEVEL 2 has LEVEL 1 as PARENT
AND REGION.OU_IP_ID = DISTRICT.PRN_OU_IP_ID --> LEVEL 3 has LEVEL 2 as PARENT
AND DISTRICT.OU_IP_ID = STORE.PRN_OU_IP_ID --> LEVEL 4 has LEVEL 3 as PARENT
AND CUSTOMERS.CL_ID = STORE.OU_TP_ID
AND STORE.OU_IP_ID IN (SELECT DISTINCT BASKET.OU_IP_ID FROM DWH.MKT_BSKT_TXN
BASKET)
```

7.6.4 *Designing the first level*

The first level of the hierarchy is the organizations level and is characterized with a null value in the parent column. A Where condition operator is holding this level of the hierarchy with a “INPUT_03.PRN_OU_IP_ID IS NULL” condition. The input port of the operator will be plug with the data set from the Organization unit.

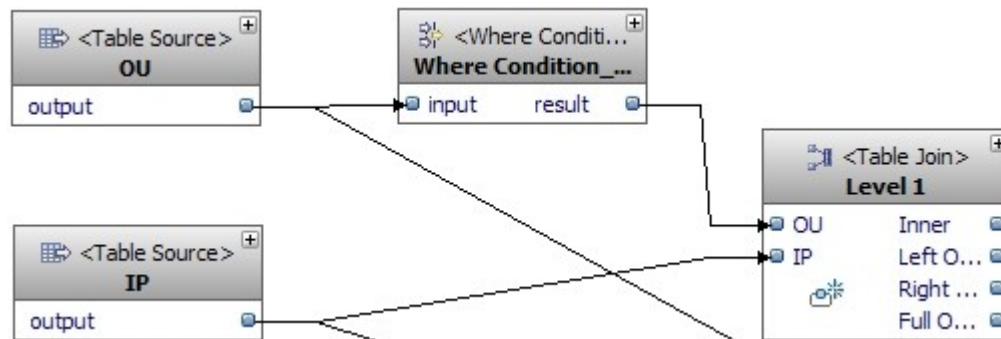


Illustration 31: Part of the data flow that get the top level of the hierarchy.

7.6.5 *Designing the other levels*

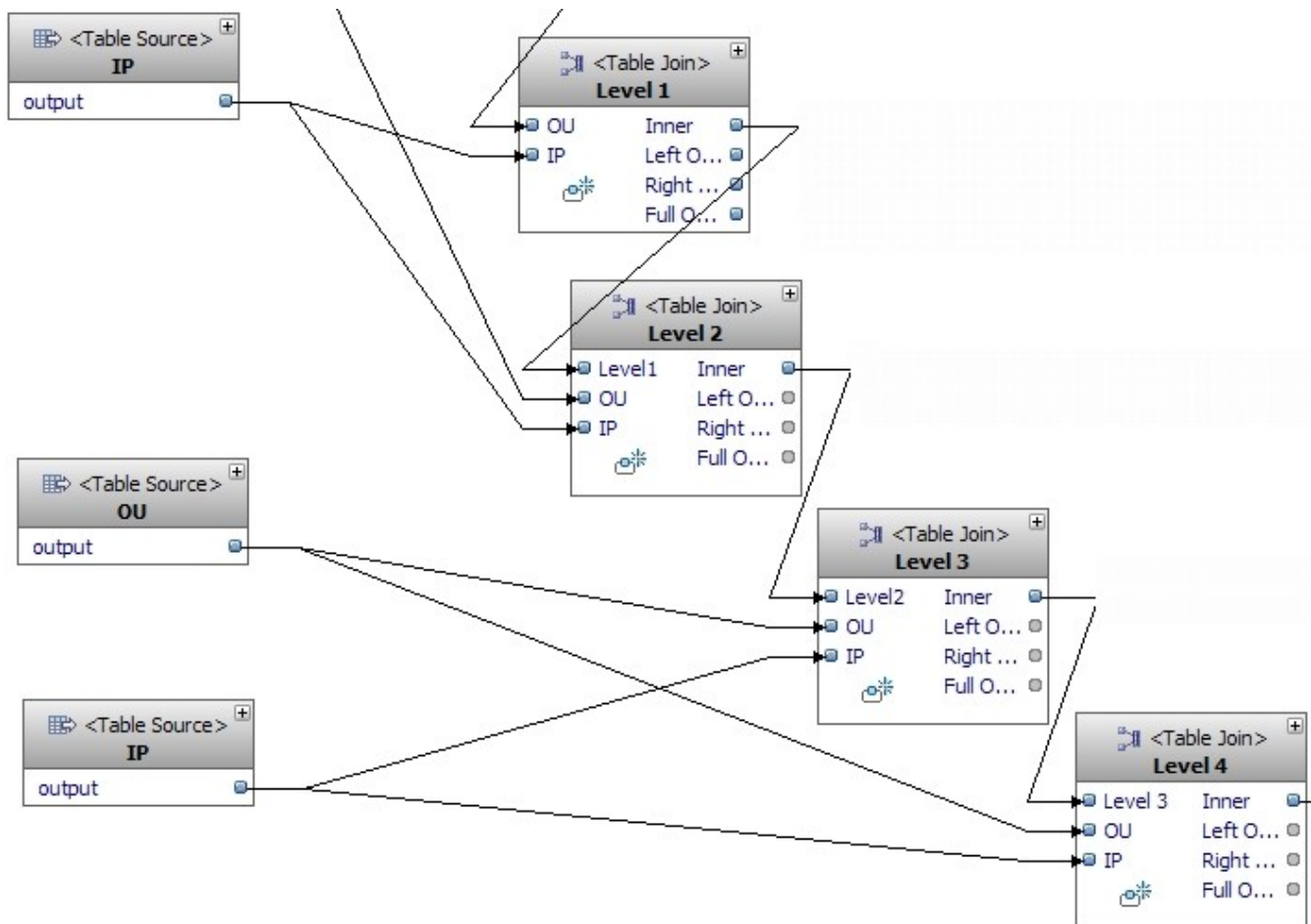
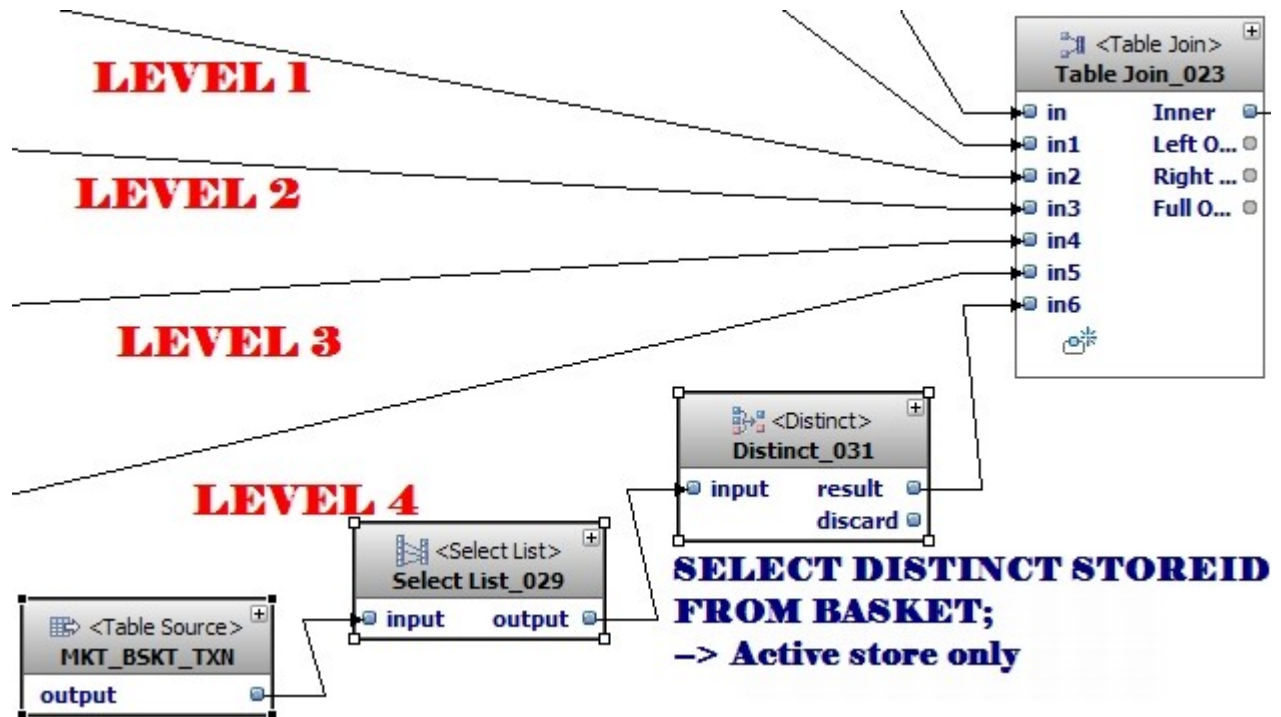


Illustration 32: From level 1 to level 4 data flow part.

The next level follow the same mechanisms but instead of holding the rows with null value parent, the rows with previous level parents are holden.

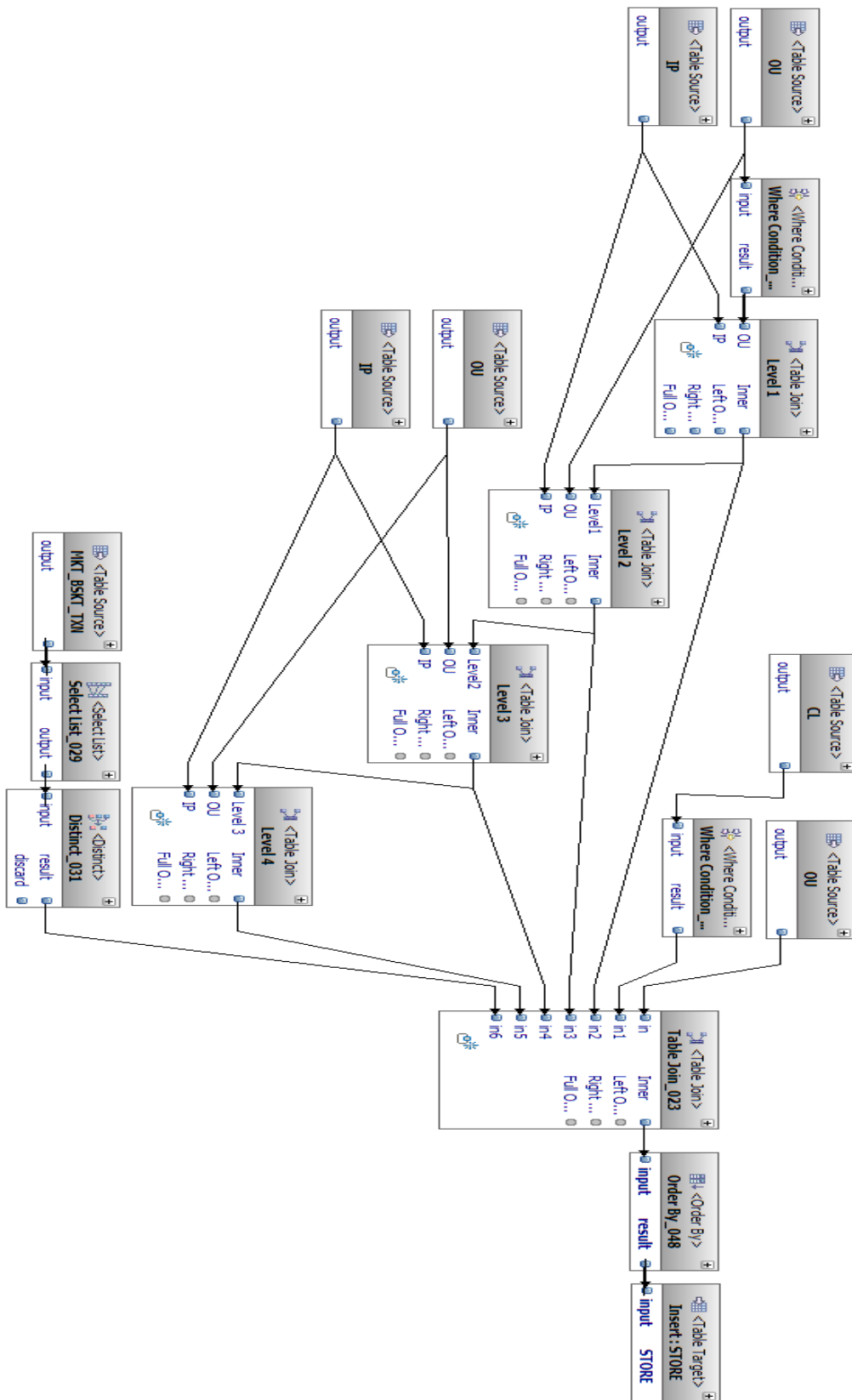
Next to the four levels, we join the four tables so that we can have all the hierarchy on the same row. In order to keep only relevant rows, we join also the levels with the Market basket so that we only keep the stores that have already sold something.



The data set from the Inner Join can then be loaded in the MARTS.STORE dimensional table using a bulk load target operator or table target operator.

7.6.6 Overview of the data flow

The complete data flow can be seen on the next page.



7.7 Populating a fact table

7.7.1 Origin and details of the data

The purchase profile analysis table is the fact table of the MARTS schema and contains the aggregated data from the Item Transaction table of the DWH schema that analysts are interested in.

The data extraction is done with usual operators. We need to join DWH tables to get the data we need and also to transform foreign key in usable data.

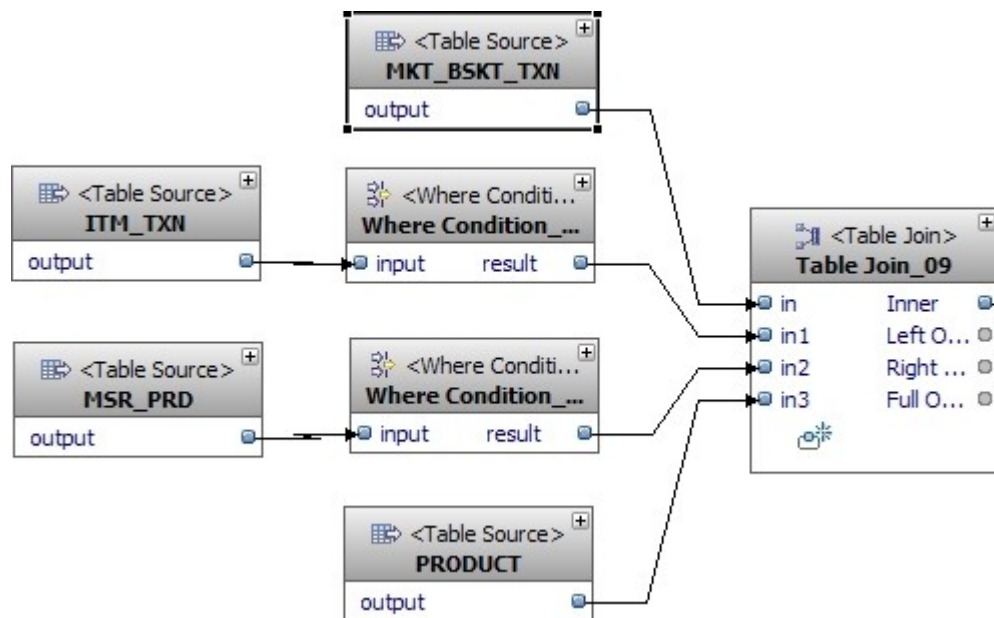


Illustration 33: Extracting data from the consolidated schema.

The condition property of the table join is set a follow:

<u>Condition</u>	<u>Comments</u>
IN_09.MKT_BSKT_TXN_ID = IN1_09.MKT_BSKT_TXN_ID	Join the Market Basket Transaction with the Item Transaction table.
IN1_09.PD_ID = IN3_09.PD_ID	Join a transaction Item with the product table to get information about the products (expected sold price,...)
DATE(IN1_09.ITM_TXN_TMS) = IN2_09.EFF_DT	The date is stored in the ITM_TXN table and the different representations of this date in the MSR_PRD table. The joined allowed us to have all the representations. (Gregorian Calendar representation, fiscal year and so on...)

Next to the previous table join operators, we start aggregating the data by using a “Group By” operator and defining the calculation of the different analyzable values in the select list properties view of the operator.

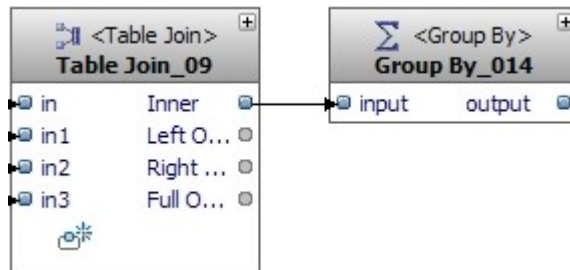


Illustration 34: Aggregation of data in the group by operator.

The Group by column are defined in the Group by tab of the property view. The group by in the example where the following:

Expression	Column Name	Data Type
INPUT_014.CNTPR_ID	CNTPR_ID	INTEGER
INPUT_014.OU_IP_ID	OU_IP_ID	INTEGER
INPUT_014.MSR_PRD_ID	MSR_PRD_ID	SMALLINT
INPUT_014.MKT_BSKT_TXN_ID	MKT_BSKT_TXN_ID	INTEGER
INPUT_014.PD_ID	PD_ID	INTEGER

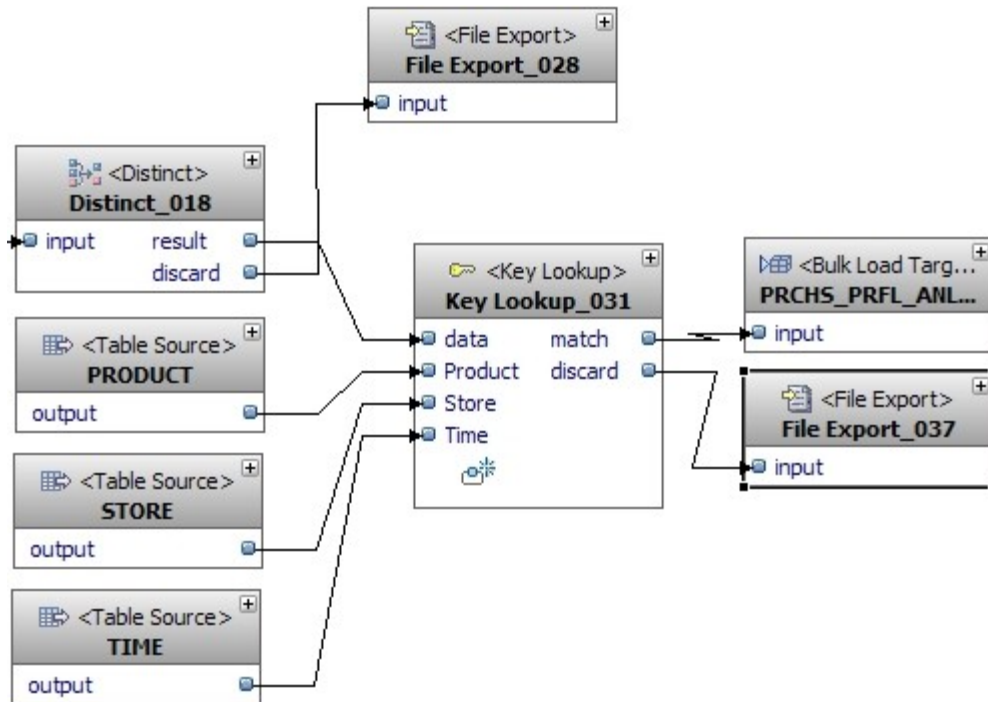
Illustration 35: Group by column listing.

The aggregated data are calculated in the Select List tab following the information in the next table:

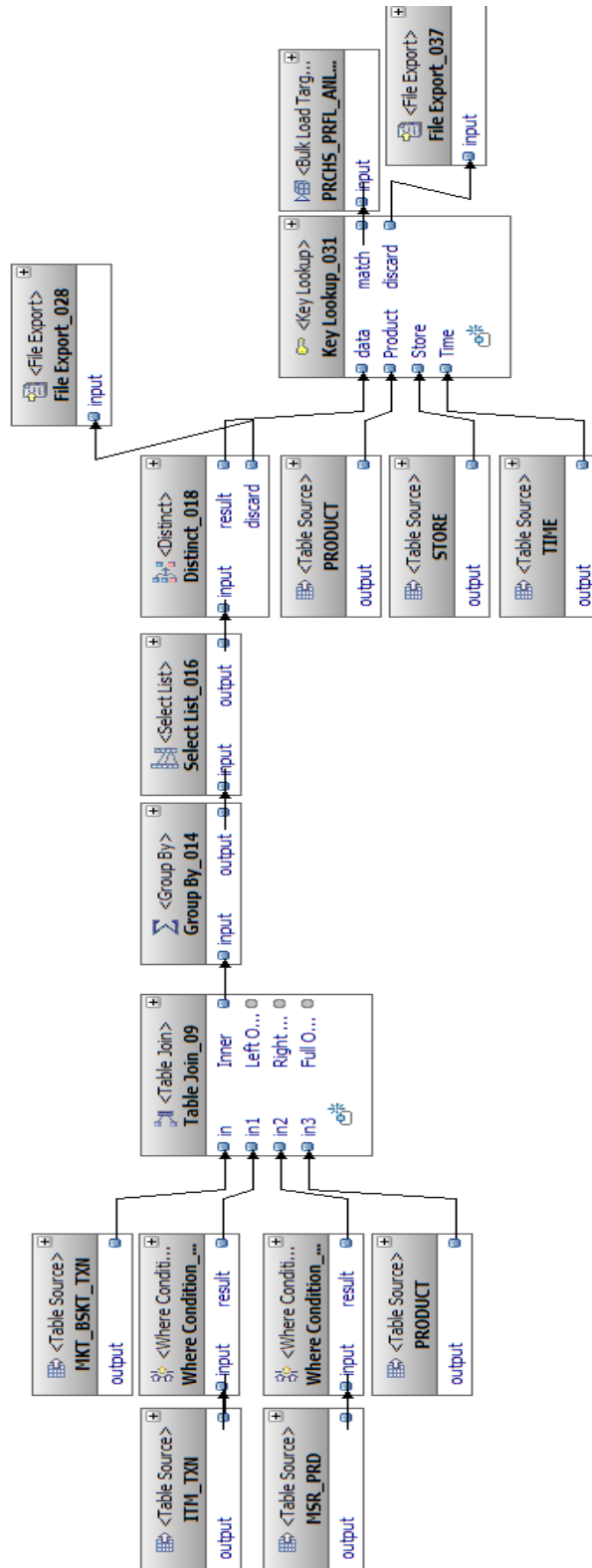
<u>Name</u>	<u>Calculation</u>	<u>Comments</u>
SUM_NBR_ITM	<code>SUM (NBR_ITM)</code>	Quantity of item in the bill.
C_D_MKT_BSKT_TXN_ID	<code>COUNT (</code> <code> DISTINCT MKT_BSKT_TXN_ID_1</code> <code>)</code>	Number of market(s) owned by the customer.
BKP_SUM_NBR_ITMXPRC	<code>SUM (</code> <code> BKP_NBR_ITM *</code> <code> BKP_ITM_STM_PRC</code> <code>)</code>	The price at which should be sell this quantity of items following catalog price .

7.7.2 Using the Key lookup operator

The fact table references data from the dimensional tables and to prevent from enforcing the referential integrity a key lookup operator is used and the three dimensional tables will be plugged with the lookup port of the operator.



7.7.3 Overview of the flow



8 CONTROL FLOWS

8.1 Definition of a control flow in Design Studio

A control flow is a graphical representation of the execution order of data flows that must follow a specific sequence. The data flows were the base of the Data warehousing applications. The second level is the control flow and is designed as data flow in the **Data warehousing project**.

In order to create a control flow in the data warehousing project, we right-click the control flow directory and we click on new Control Flow. A new control flow is added into the directory. A control flow can be into another project than the one for the data flow but to have a good project management and to make it easy to create data warehousing application, it is better to use the same project.

8.2 Rules and operators from control flows

8.2.1 Rules

Before designing control flows, some rules must be known. First of all, each control flow must have at maximum one starting point. This starting point is represented by a start operator that represents the entrance point of the control flow process.

In the opposite, there will be a lot of end points that represent the end of the process. These ending points are modeled with an End operator. A control flow is a kind of horizontal tree where there is one root, the start point and where each leaf must be an end operator.

8.2.2 Outputs

Most operators have outputs. The sequence goes from one operator to another using these outputs. These outputs define what must be done if the execution of the operators was successful or if errors occur.

The **V output** is the on success output. If the process leaves the operator from this output, it means that no error occurred.

The **X output** is the on failure output. It is used to catch exception.

The **--> output** is the Unconditional output. If something is plug in this output the two other outputs are not available anymore.

8.2.3 Definition of available control flow operators

8.2.3.1 Start operator



Start

An all new created control flow is not an empty canvas but it contains a Start operator. This operator will be the entrance point in the process. This operator must be unique in the control flow as explained in the “9.2.1 Rules”.

The blue arrow, the red cross and the clean up tool are the three outputs of the start operator. Next to the blue arrow, the first custom operators are plugged.

The red cross is the “On failure” output. At this part of the control, no error may occur but if a future operator does not catch an exception, the exception will be thrown to the previous operator until the exception is managed or until the exception ends to the Start operator which is the last operator that can manage it. An operator catches an exception if operators are connected to the “On failure” output of this one.

The clean up output is executed at the end of the control flow to clean up. For example, if temporally files are created, we could remove these files here. In the other hand, the code next to the “Clean up” output is executed no matter if the control flow ran until the end or got interrupted because of an exception.

8.2.3.2 End operator



Operator.

The end operator is the last operator for each branch. This operator results in stopping the process and executing the “Clean up” output of the Start

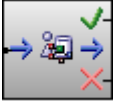
8.2.3.3 Fail operator



failure” output.

The fail operator explicitly causes the control flow to throw an exception and forced the execution to return to last operator that managed the “On

8.2.3.4 Data Flow operator



The data flow operator call a data flow. Thanks to this operator, it is possible to move, transform and load data following a previously designed data flow.

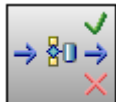
In data warehousing application, the data flow consists in the base level of the application. And the calls of these data flows in a control flow is the second level in the development of a data warehousing application.

8.2.3.5 File Wait operator



The file wait operator checks the existence of a file. This is interesting to avoid proceeding a data flow that load data from a file, if this file does not exist. This operator is also used to check if a connection to an external database is available or not which is essential if the process attempts to replicate data.

8.2.3.6 Stored Procedure operator



This operator calls a procedure stored in the database.

Note: A control flow is designed into a Data Warehousing Project. These projects are based on a Data Design Project that consists of a data model. To enable the operator to retrieve the stored procedure listing, we have to make sure that during the reverse engineering process, the ad hoc data base elements have been selected.

8.2.3.7 Variable Comparison operator

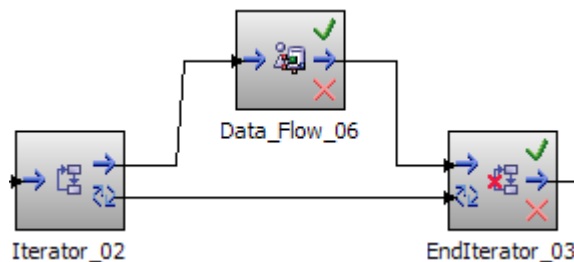


The variable comparison operator compare variable values. The comparison type can be numeric, string , null or boolean.

<u>Comparison type</u>	
Numeric	= , !=, <, >, <=, >=
String	Equals, Not Equals, Sub String of, Contains, Start with
Boolean	Is true / is False
Null	Is Null / Not Null

The output of the operator are little different from the others but are explicit enough. The + is the true output and the - the false output.

8.2.3.8 Iterative processing loop



This operator brings two objects on the canvas. An Iterator and an EndIterator surround the code in the loop.

The number of times the loop must be performed is defined in the property view of the operators and can be set to three different modes.

The modes are “fixed number of times”, “repeat loop for each delimited data item in a file” and “repeat loop for each file in a directory”.

In our study case, the second mode could be very interesting for example if we could get from the file the information to connect to the different databases of the stores and repeat the same data flow which will have as parameter the connection information to the data base of the store. If a new store appears, the store is added in the file and the data warehousing application will extract the data from the new store also (No need to redeploy the application, only the configuration file is changed).

8.2.3.9 Email operator

This operator is used to send a mail that may alarm the administrator of the the use of branch in a control flow. For example, if an error occur during a data flow operator. It could be possible to send a mail to warn the administrator that there was a problem.

In the other hand, errors can be also traced from the log files of the Websphere application server.

8.3 Using the control flow to feed the MARTS schema

8.3.1 Process understanding

In the previous chapter called “Populating the Item Transaction table”, we have designed a data flow that loads the sales and refunds from the stores into the consolidated database. We should want to run this data flow first.

When the consolidated schema is filled, the MARTS schema is still not updated. In the previous chapter called “Populating a dimensional table of the marts schema”, we have designed data flows that Extracts data from the consolidated schema, Transform and Load the data in the dimensional table. In fact we have one data flow for each dimensional table. It consists in three data flows that are ETL tools for the PRODUCT, TIME and STORE tables.

In the end the fact table is loaded with the data flow from the chapter named “Populating a fact table”. From this moment, we have created different bricks (data flows) and now we want to build the wall (control flow) which is the main component of the house (Data warehousing application). In our case, the bricks must be dropped following a specific order.

Indeed the fact table must be loaded after the dimensional tables that must be also loaded after the consolidated schema have been updated. This sequence of execution is designed in a the control flow.

8.3.2 Designing the data flows

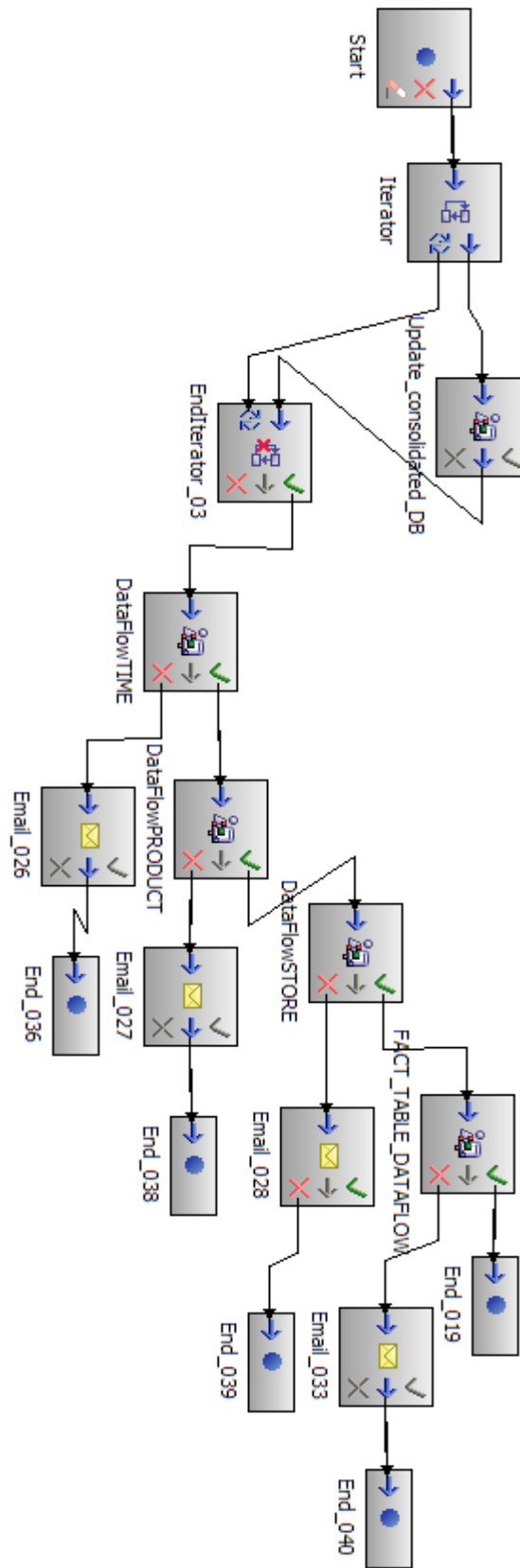
The control flow order the execution of the five data flows using five data flow operators. If an error occur during the execution of one of the data flow, an email will be send to the administrator to allow him to check and resolve the problem. The email is send using the email operator. This operator works only if the control flow is deployed on a Websphere application server and if the server have a SMTP server configured.

The order of execution is :

1. Loading the consolidated database with data from stores.
2. Loading the dimensional tables : STORES, PRODUCTS and TIME.
3. Loading the fact table

The first data flow have been also changed so that it receives a parameter which corresponds to a connection to a store database. So the first data flow is surrounded by iterator operators and is run as many times as they are stores. We don't have for example, one data flow for each store but a customizable data flow thanks to a parameter.

A illustration of the complete control flow is available on the next page.



9 DATA WAREHOUSING APPLICATIONS

9.1 Introduction

The Data warehousing is the finality of all we have done until now. This kind of applications are used to build a data warehouse. At this point, we just have designed control flows that contains and orders data flows. The data warehousing application is the way to concretize, to give a life to the graphics we designed previously.

The first part of the work will be in Design Studio where the Data warehousing project will be transformed into a Data warehousing profile. This profile is the application that will be deployed on the application server.

The Websphere application server is the platform used to run data warehousing applications. DWE administration console allows to deploy, undeploy, run and manages the applications on the Web Sphere application server.

The second part will be to create a resource, so that the Websphere can connect to the IBM DB2 UDB and then to deploy the application on the server.

9.2 Deployment preparation

In Design Studio, we have created a data warehousing project to design flows. The data warehousing application builds the data warehouse by executing one or more control flows.

The data warehousing project provides a wizard to create an application profile. The creation of a profile is divided in two parts. The first one is the selection of the control flows that must be run by the application and the second part is the generation of the ETL code that is packed in a **zip file**.

In order to create a new data warehousing application, a right-click on the data warehousing project and on the new button is enough to access the wizard.

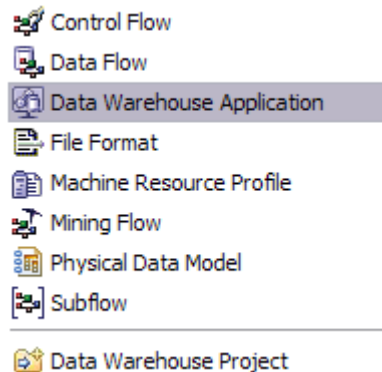


Illustration 36: Starting the wizard.

The wizard is easy to use and developers have only to choose the control flow and configure some settings like the name of the profile, the directory to save the package (zip file).

9.3 Working with DWE Administrating console

9.3.1 Prerequisites before deploying the application

The DWE Administrating console is a Websphere application which is accessible from the browser. The default URL to connect to the console is <http://theappsvr:9080/dweadm>.

But first, the Websphere application server must be started. The default instance of Websphere is called server1. This instance is start with the command :

```
startserver server1
```

The “startserver” executable is located in the bin directory of the Websphere one. When the server is started, we can connect to the Administrating console.

In the SQL warehousing directory in the left frame, we have access to all the authorized actions. The available actions depend on the user authorizations.

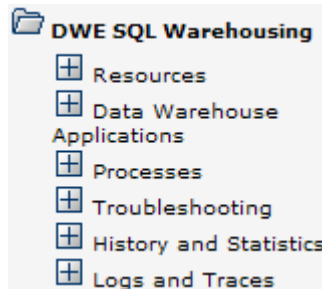


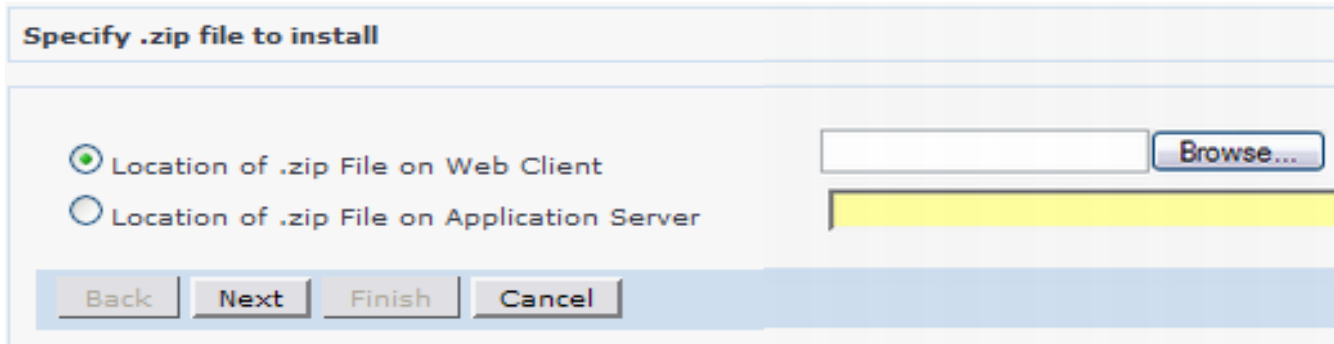
Illustration 37: Available actions with db2admin login.

From the browser, we can navigate the menu to add a data source that are required to the application to run and to Websphere to connect to the database. (Resources -> Add Data Sources)

NOTE: Websphere and Design Studio have both the SQL Warehousing component which can transforms the control flows and data flows into SQL query. In fact the control flows and data flows are saved in XML files which are parsed by the SQL Warehousing that does XML Transformations to get the SQL queries.

9.3.2 *Deploying the application*

Now that Websphere and DWE Administration Console are running, the application can be deployed. The first step is to upload the **zip file** on the Websphere application.



The screenshot shows a dialog box titled "Specify .zip file to install". It has two radio button options: "Location of .zip File on Web Client" (which is selected) and "Location of .zip File on Application Server". To the right of the first option is a text input field and a "Browse..." button. Below the second option is a yellow highlighted area. At the bottom, there are four buttons: "Back", "Next", "Finish", and "Cancel".

The file is uploaded the same way file are attached to a mail if using a web-mail.

The next steps before ending the deployment can be let the default. For the information, the latter steps allow to change the level of tracing and the places where will be located the log and temporary files.

When the application is deployed, it does not mean the application is running. From the processes directory in the left menu, the management of deployed application is available. From there it is possible to define the scheduling for the execution of the deployed applications.

9.4 JK Superstore builds the Data warehouse

JK Superstore needs one data warehousing application. This application must execute the control flow that we have created previously everyday at 12PM to get daily sales and refunds. Thanks to this application, the consolidated database is updated automatically and the analyst can have access to the Business Information without having to check if the database is loaded with latest data.

At this part of the project, the data warehouse is completed and is automatically updated. Days after days the size of the warehouse increases and the data mining could be a good way for the analyst to find knowledge in the data. In the next chapter, we start the data mining to help the analysts to make decisions by finding knowledge he had maybe not find himself because of the amount of data.

10 DATA MINING

10.1 What is the data mining?

The data mining is a part of the famous Knowledge in Database Discovery process. The data mining correspond to the discovery part of the process. The aim is too discover unknown knowledge from an important amount of data.

A lot of definitions have been done and here is some of the most famous one :

- “Data mining is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data” from W. Frawley in the book called “Knowledge Discovery in Databases: An Overview”⁶.
- “Data mining is the science of extracting useful information from large data sets or databases “ from D. Hand in his book named “Principles of Data Mining”⁷.

10.2 Why using data mining?

Utilizing the data mining in I.B.M. Data Warehouse Edition was the second part of my project. In the first part, a data warehouse has been built and contains now an increasing amount of data. The product table contains more than 38.000 products and an half million of transactions. I have decided to test the data mining on this big amount of data.

I have decided to do data mining on the JK Superstore database because the Market Basket database is the best example to try to find Association Rules. In other words, we want to know what a customer could buy if he buy a product. This can improve the benefit of JK Superstore because thanks to the data mining results they will be able to change for example, the place of the products in the stores.

After that we will try to find clusters in the titanic data set. I have also used this data set with the decision trees also called classification in data mining.

6 ISSN (International Standard Serial Number) : [0738-4602](#)

7 [ISBN 0-262-08290-X](#)

10.3 Association rules

The explanations are not exhaustive but allow to have a enough knowledge to understand the data mining results.

The association rules are found patterns after a data mining process. An association is a relationship between items. A rule is a formal means for representing an association which consists in a rule head and a rule body.

Example: A association rule

[Beers] ==> [Diaper]

[Beers] is the body of the rules and [Diaper] is the head. This rule means that if someone buy beers, this person buy in the same time diaper.

These information are interesting but does not allow to make decisions because we need to know if the rule is relevant or not. For example, it could be interesting to know what is the support, the confidence and the lift of this rule.

The support is the probability that a transaction contains the rule head and rule body.

Calculation: The Support of a rule.

$$\text{Support}(\{\text{Beers}\} \rightarrow \{\text{Diapers}\}) = \frac{\# \text{ of transaction containing } \{\text{beers, diaper}\}}{\text{total } \# \text{ of transactions}}$$

The confidence is the conditional probability that a transaction which contains the rule body also contains the rule head.

Calculation: The confidence of a rule.

$$\text{Confidence} (\{\text{Beers}\} \rightarrow \{\text{Diaper}\}) = \frac{\# \text{ of transaction containing } \{\text{Beers}\}, \{\text{Diaper}\}}{\# \text{ of transaction containing } \{\text{beers}\}}$$

Example :

<u>Transaction Id</u>	<u>Bought item(s)</u>
1	A,B,C
2	A,C
3	A,D
4	B,E,F

If we work with the item set $X = \{A,C\}$, we can have two rules which are A->C and C-A.

	<u>Support</u>	<u>Confidence</u>
<u>A-C</u>	50%	66,7%
<u>C-A</u>	50%	100%

The lift gives the importance of a rule⁸.

Calculation: The lift of a rule.

$$\text{Lift} (\{Beers\} \rightarrow \{Diaper\}) = \frac{\text{Confidence} (\{Beers\}, \{Diaper\})}{\text{Support} (\{Diaper\})}$$

⁸ Some rules may have the same lift but not the same importance, in this case the subtractive lift calculated.

10.4 Easy mining procedures

10.4.1 Introduction to Easy Mining

The data mining with I.B.M DB2 is done thanks to stored procedures. These easy mining procedures create models that are saved on the database.

A model is saved on the database if the stored procedure created a view and add the model in the model list table. In the model list table, the mining results is saved in a PMML file. A PMML file is The Predictive Model Markup Language. This file is written in XML-based language. It allows to share mining results between different platforms that are compliant with PMML. For example, we could use it to share mining results between heterogeneous RDBMS (Oracle and I.B.M. DB2).

Sample: Part of a PMML file generated with the titanic data set

```

=<DataDictionary numberOfFields="4">
=<DataField name="SOCIAL" displayName="SOCIAL" optype="categorical">
  <Value value="1st" /> <Value value="2st" /> <Value value="3st" /> <Value
value="crew" />
  </DataField>
=<DataField name="ADULT" displayName="ADULT" optype="categorical">
  <Value value="Adult" />
  <Value value="Child" />
  </DataField>
=<DataField name="SEXE" displayName="SEXE" optype="categorical">
  <Value value="Male" />
  <Value value="Female" />
  </DataField>
=<DataField name="STATUS" displayName="STATUS" optype="categorical">
  <Value value="S" />
  <Value value="D" />
  </DataField>
</DataDictionary>
=<ClusteringModel modelName="TEST.TITACLUSTB modelClass="distributionBased"
functionName="clustering" algorithmName="Demographic" numberOfClusters="3"

```

10.4.2 FindRules

The FindRules procedure is used to find association rules, relationship between items. This is the one, that we use in the JK-Superstore.

Syntax: FindRules Syntax

```
IDMMX.FindRules(<rulesView>,  
                <inputTable>,  
                <groupColumn>,  
                <nbRules>,  
                <minConfidence>)
```

The <rulesView> parameter is the name of the view that the function must create. (A model correspond to a PMML file and a view).

The <inputTable> correspond to the table source for the data mining. The columns of the table must be divided in two groups. One group corresponds to the <groupColumn>. The groupColumn is the transaction identifier. The columns of the table that are not specified in the groupcolumn are considerate as the “Items” columns.

The <nbRules> parameter defines the minimum rules the data mining must return.

The <minConfidence> parameter is the minimum confidence that the rules must return.

10.4.3 ClusterTable

The ClusterTable is used to find groups with similar characteristics. These groups are called segments or cluster.

Syntax: ClusteTable procedure

```
IDMMX.ClusterTable(<clusterView>,  
                  <inputTable>,  
                  <minSize>, -- Minimum size of a cluster  
                  <maxSize>) -- Maximum size of a cluster
```

10.4.4 *The others functions*

During the project, I have been using almost all the easy mining procedures. In this report, I will limit to two of them but here is the complete list⁹ of easy mining procedures the intelligent miner from DB2 Data Warehouse Edition provides:

Mining tasks	Easy Mining procedure
Finding deviations	FindDeviations
Finding groups with similar characteristics	ClusterTable
Finding relationships	FindRules
Finding sequential rules	FindSeqRules
Predicting future behavior	PredictColumn
Predicting an outcome	PredictColValue
Finding explanations for specific events	ExplainColValue
Finding most important fields	FindMostImpFields

Illustration 38: Easy mining procedures

⁹ The complete list and documentation can be found here :
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.im.overview.doc/c_intro_em_procs.html

10.5 Association rules on the Market Basket of JK Superstore

10.5.1 Introduction

The JK Superstore consolidated database contains a lot of market baskets. In the beginning a was looking for rules on the products but this is not efficient. There is a problem with the different brands of items. Because if we buy “Coca cola” and “Chips” and another person buy “Pepsi” and “Chips” no rule can be found. The grain of mining must be quite selected.

In The JK Superstore, we are going to use the GRP table to regroup items in a same category. So that for example “Coca cola” and “Pepsi” are both in the category “Soda in Without alcohol in Drinks”.

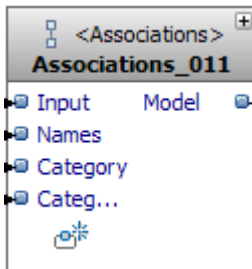
The data mining will be designed into a mining flow which cleans the data (regroup the items in categories). A mining flow can be created in the same project than the control and data flows. We could use the same Data warehousing project that we have used to Design the application previously.

Note and problem I had during my project : By default the Intelligent Miner is not enabled on a database. If we try to call an easy mining procedure without the IM enabled, an exception occurs. To activate the Intelligent Miner, the following command must be executed:

```
idmenabledb databasename fenced dbcfg
```

10.5.2 Cleaning the data and Creating a model

Using a mining flow is almost the same than using a data flow. The difference is that the mining one have data mining operators in addition. It exists one mining operator per easy mining procedure because a data mining operator will be transformed into a SQL query which call the easy mining procedure thanks to the SQL Warehousing tool.

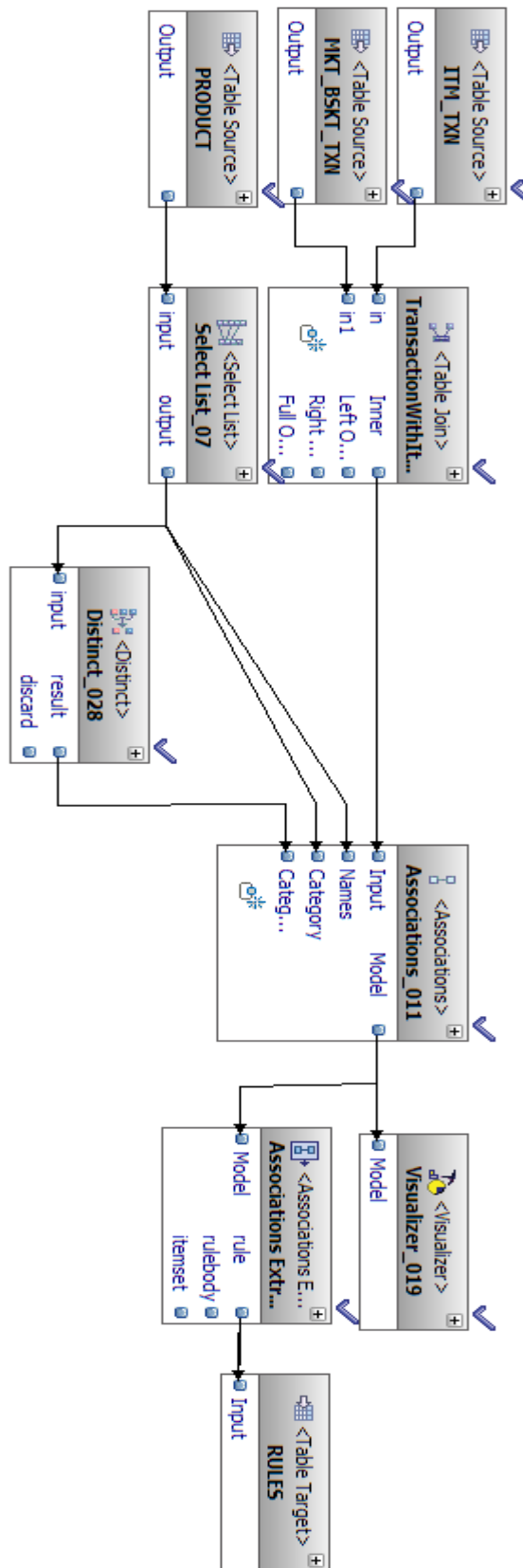


Our goal is to find relationships between products and more accurately thanks to the category of the product and it will be realized with an associations operators. This operator have three inputs: one for the data that must be logically divided in two groups the Transaction Id and the Item Id, the name input for the mapping between item id and it name and a category input to define the links between a product and a category.

We have to clean the data so that we can have a view with columns. One with the market basket id and the other with the product id. We will also need the product table, to do the mapping of the name the product.

The output of the associations operator is the created model. This model will be loaded in the visualizer to get the result but it will be also used with a rules extractor that will fill the view with the rules.

The complete data flow is available on the next page.



10.5.3 The visualizer

The Intelligent Miner Visualizer is a software that is installed with the Client components of IBM Data Warehousing Applications. This tool is able to connect to a database and shows the data mining models available on this one.

The visualizer provides a graphical representation of the model. It is also possible to download the PMML file, to export the model to a file.

We are going to use this to interpret the result of the data mining on the JK-Superstore.

10.5.4 Interpreting the result of the data mining

When we run the visualizer, no model are open. We directly arrive in front of the panel that allows to connect to a database, get the list of models and open a model.

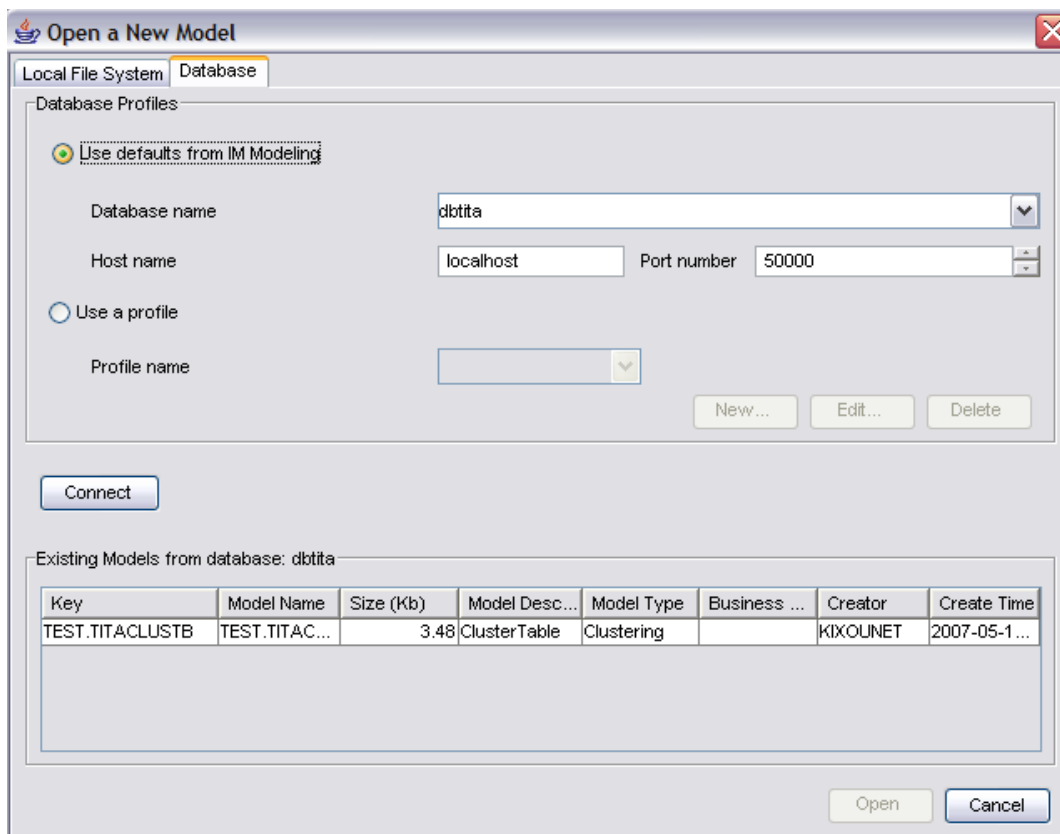


Illustration 39: First screen of Visualizer

When the model is opened, three tabs allows to browse in the visualizer. The first tab is the rules tab which shows the rules found by the intelligent miner.

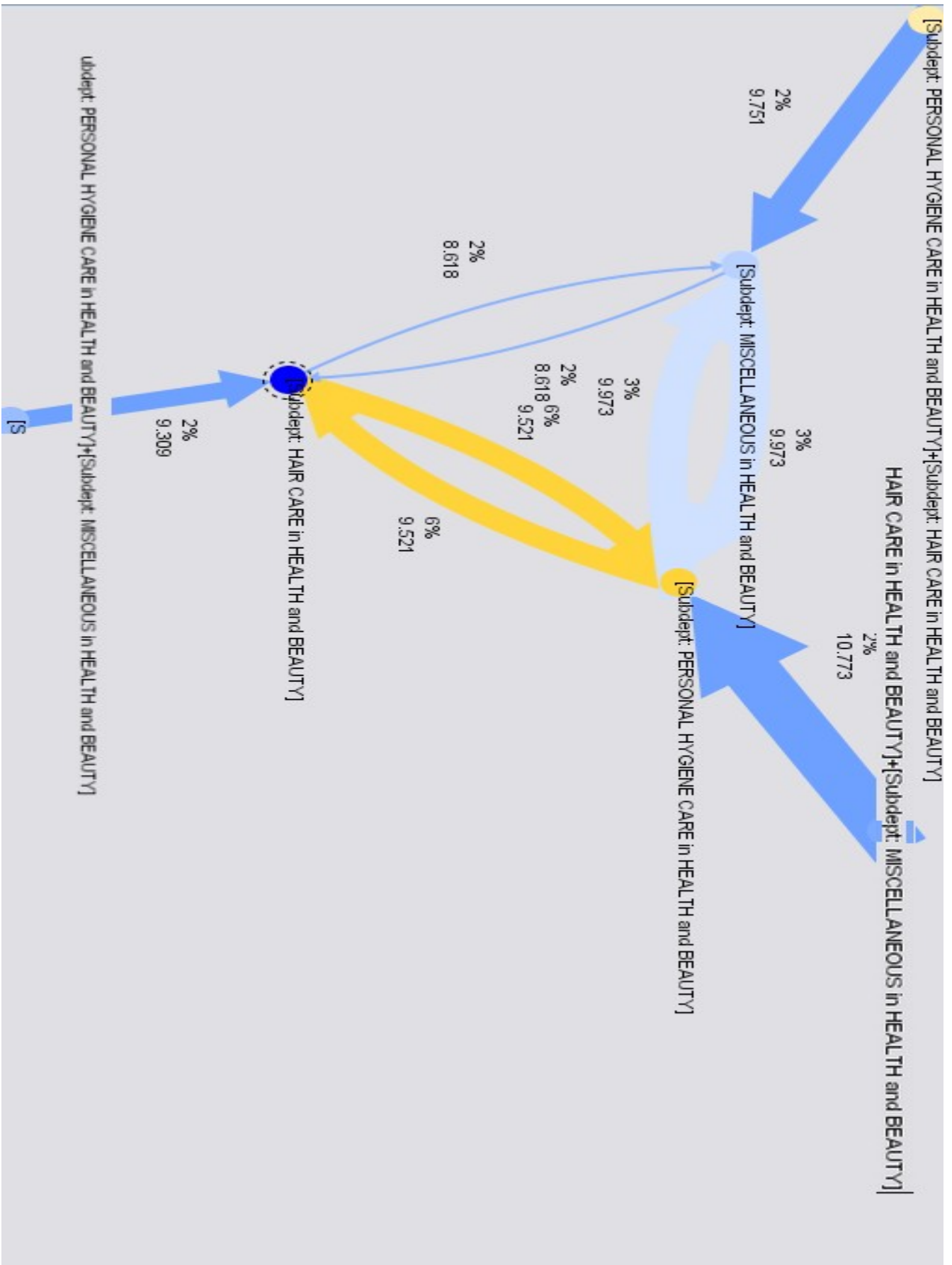
The next tab in the visualizer is called “Item sets”. The items set are listed and the support of the item set is given. We also get the number of times the item set exists as rule head and rules body.

Here is a sample of the found rules:

Rule	Support	Confidence	Lift	Absolute Support	Subtractive Lift
[Subdept: HAIR CARE in HEALTH and BEAUTY]+[Subdept: MISCELLANEOUS in HEALTH and BEAUTY] ==> [Subdept: PERSONAL HYGIENE CARE in HEALTH and BEAUTY]	2.2053%	90.2238%	10.77	766	0.82
[Subdept: MISCELLANEOUS in HEALTH and BEAUTY] ==> [Subdept: PERSONAL HYGIENE CARE in HEALTH and BEAUTY]	2.9769%	83.5218%	9.97	1,034	0.75
[Subdept: HAIR CARE in HEALTH and BEAUTY] ==> [Subdept: PERSONAL HYGIENE CARE in HEALTH and BEAUTY]	6.3454%	79.7395%	9.52	2,204	0.71
[Subdept: PERSONAL HYGIENE CARE in HEALTH and BEAUTY] ==> [Subdept: HAIR CARE in HEALTH and BEAUTY]	6.3454%	75.7649%	9.52	2,204	0.68
[Subdept: PERSONAL HYGIENE CARE in HEALTH and BEAUTY]+[Subdept: MISCELLANEOUS in HEALTH and BEAUTY] ==> [Subdept: HAIR CARE in HEALTH and BEAUTY]	2.2053%	74.0812%	9.31	766	0.66

Illustration 40: Relationships found after data mining on the market baskets of JK Superstore

The third tab is a graphical representation of the rules. The items are dots and the associations are arrows. The lift is representation by the size of the arrow and the color represent the support. Available on the next page.



The rule with the highest confidence:

At this point, we have access with the visualizer at all the static information found by the intelligent miner but we have now to understand and interpret the result.

When people buy items from the “Hair care” category and items from the Miscellaneous (which is sub category of Health and beauty), people also buy items from the Personal Hygiene in 90% of the cases.

This information is very interesting because of a high confidence of 90 %. It means that almost everybody buy items from both categories.

In the other hand, the support is about 2%. It means that 2% of the persons buy items in “Hair care” and “Miscellaneous” together. This Support is low, but it must be compared with the other support. The average support of rules is between 2% and 3%. The lift shows us if the rule is relevant and this rule have also the highest lift. The lift allows us to know if a lot of other rules have the same rule head (the same consequent).

The lift has a value of ten and it is the highest lift in the found patterns. We can notice it on the previous page with the graphical result that shows the biggest arrow for this rule compare with the other rules.

It is possible in Visualizer to isolate one or more rules and to show only on one or more rules on the graph:

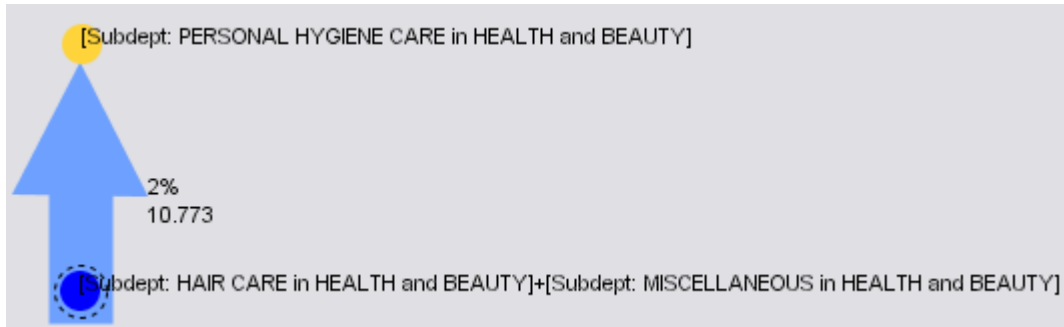


Illustration 41: Isolated rule from visualizer.

The yellow and blue circles represent the support of the item set.

The color of the arrow shows the support of the rule.

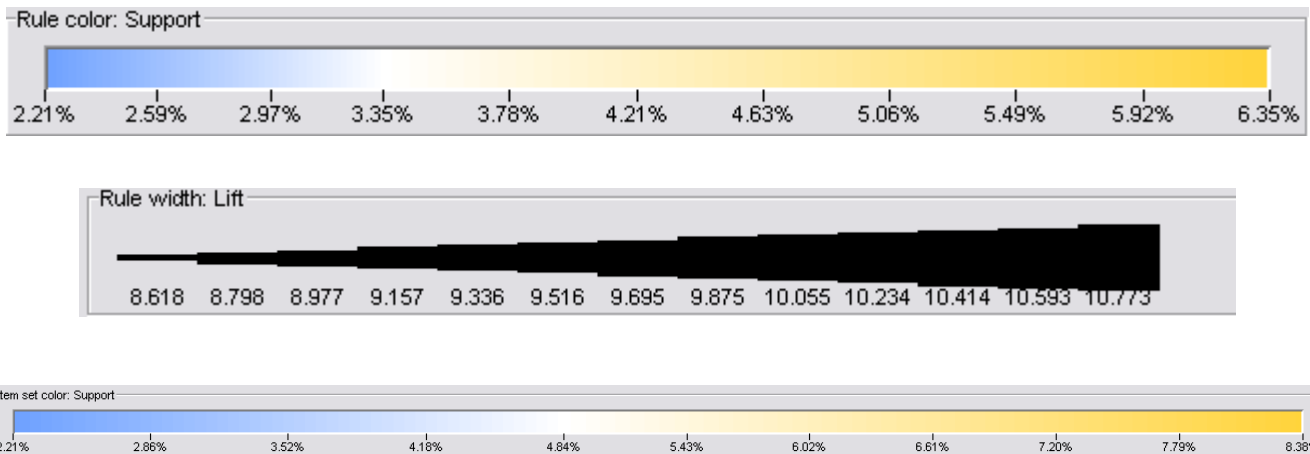


Illustration 42: Item Set Color Support.

10.6 Clustering on the titanic data set

10.6.1 Introduction

Data mining will be done on the data set of the titanic. The aim of the data mining will be to found clusters, to regroup victims in different segments depending on some characteristics. The model will be created from an Java application using a Java bean. The clusters will be interpreted from the visualizer. We will try see if all clusters respect the rule “women and children before the men”.

10.6.2 The origin of the data

The use data set has been found on Internet¹⁰. The data set contains the real information about the passengers. The titanic dataset gives the values of four categorical attributes for each of the 2201 people on board the Titanic when it struck an iceberg and sank. The categories are social class, age, sex and the survival status.

The name of the value of some columns has been changed so that it is easier to read the result. The social class has been renamed like “1st”, ”2nd”, 3rd” and “Crew” instead of 1,2,3 and 4. The Age column contains values like “Child” or “Adult”. The Sex one has values like “M” or “F” and survived status is set to “S” for survived or “D” for death.

The data set has been inserted into a a table using the CVS import tool.

<u>Script to create the database.</u>	<u>Content of the createTables.db2</u>
<pre>@ECHO off ECHO Data mining Test 1 Namotte Cédric ECHO Create the DMTEST DB db2 terminate db2 force applications all db2 drop db DMTITA db2 create db DMTITA idmenabledb DMTITA fenced dbcfg db2 connect to DMTITA ECHO Created sucessfully... ECHO Creating table(s) and loading data... db2 -tvf createTables.db2</pre>	<pre>CREATE TABLE TEST.TITANIC(social CHAR(4), adult CHAR(5), sex CHAR(6), status CHAR(1))not logged initially; import from titanic.csv of del modified by CHARDEL"" COLDEL, DECPT. insert into test.titanic;</pre>

¹⁰ <http://www.cs.utoronto.ca/~delve/data/titanic/desc.html>

10.6.3 How to create a model without mining flow.

This time, we are not going to use the mining flow to build the mining model. The model will be created using a Java bean. The bean will load the TITANIC table and call the IDMMX function. The Java code is available in the appendices but here some explanation of the Java Bean.

- All the SQL Queries are stacked into a Vector in the Constructor.
`sqlStatements.add(sqlStmt.toString());`
 - The `sqlStmt` at this moment contains: The creation of the temporally view, the Call of the data mining procedure and the creation of a new view which contains the data mining results.
- The “Execute” procedure executes the queries on the database.
public void execute(String jndiName) **throws** SQLException, NamingException

There are two algorithms in Intelligent Miner which allow to find clusters in data sets. The algorithms are the Geometric and the Kohonen. The Geometric one found three clusters and the kohonen find eight clusters in ten passes.

10.6.4 *Interpretation of the results*

10.6.4.1 *The results with the geometric algorithm*

The geometric algorithm found 3 clusters. The smallest one represent 3,36 % of the population on board.

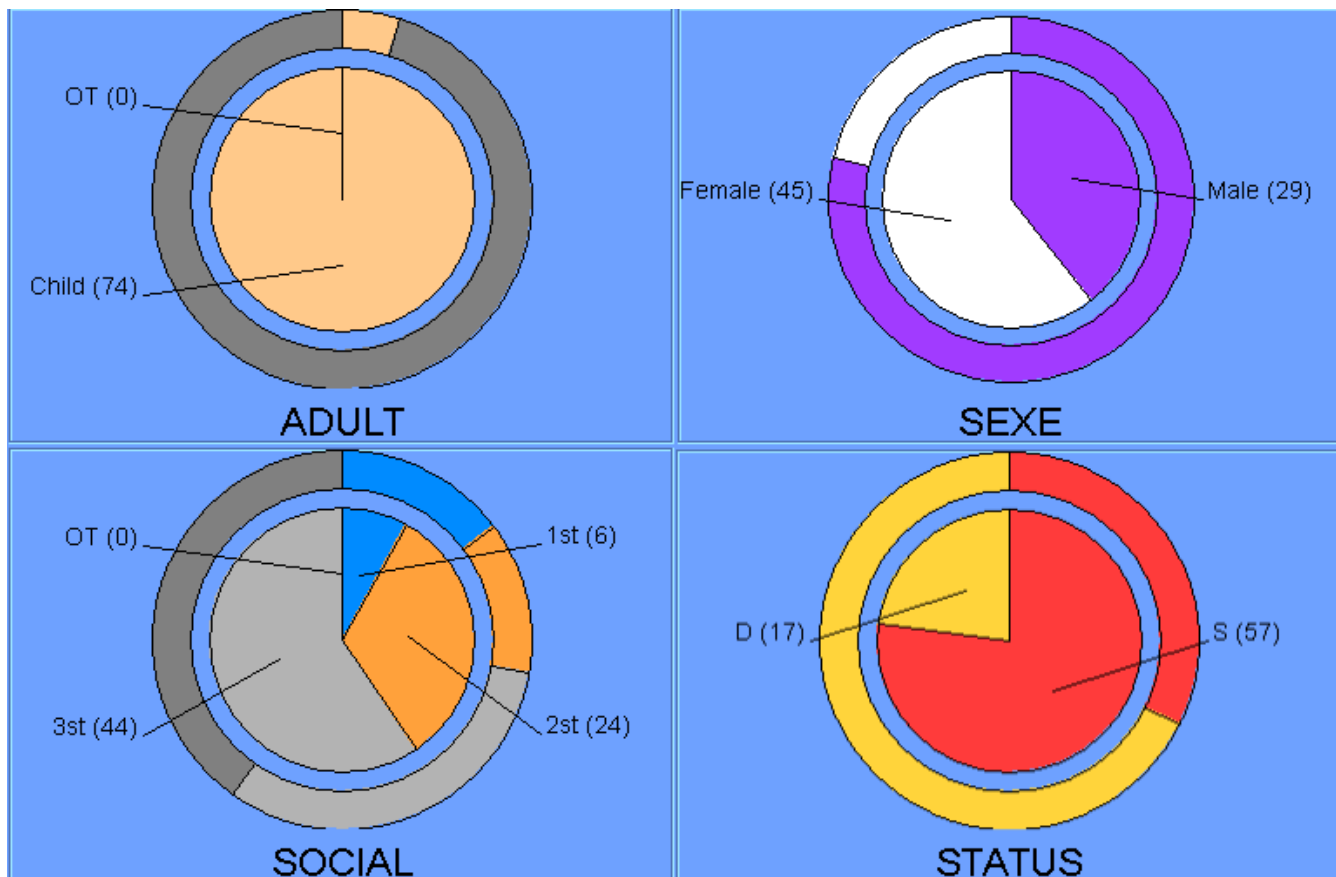


Illustration 43: Smallest Cluster found with the geometric algorithm

All graphics contains one disc and one circle around it. The inside disc is the cluster population. The outside circle is the whole population.

This cluster contains only children. There are 74 children and the whole population contains 109 children. This cluster is mainly a representation of the children in the Titanic. The frequency of children in this cluster is 100% which is a big contrast compare to the whole population. We notice also that in the children, they were more female than in the adults.

In the social category, we have a lot of information. The dark gray color represents the crew. This cluster contains no dark gray part. It means and it is normal that no children where in the crew.

One more interesting thing, the social status shows also that the parents from the third class travel more with their children than with the passengers from the first class. (I thing this is because of a demographic way of life, poor persons had more children than the rich persons).

From the information in this cluster, we can retrieve a very good information. 77,3 % of the children survived.(77,3% of the children in the cluster which represent 69% of the children on board). 77,3% is a lot compared to the whole population where 66,7 % did not survived.

This cluster proves a part of the rule: “Children before men”.

The second cluster will help us to prove that men on Titanic were *gentlemen*.

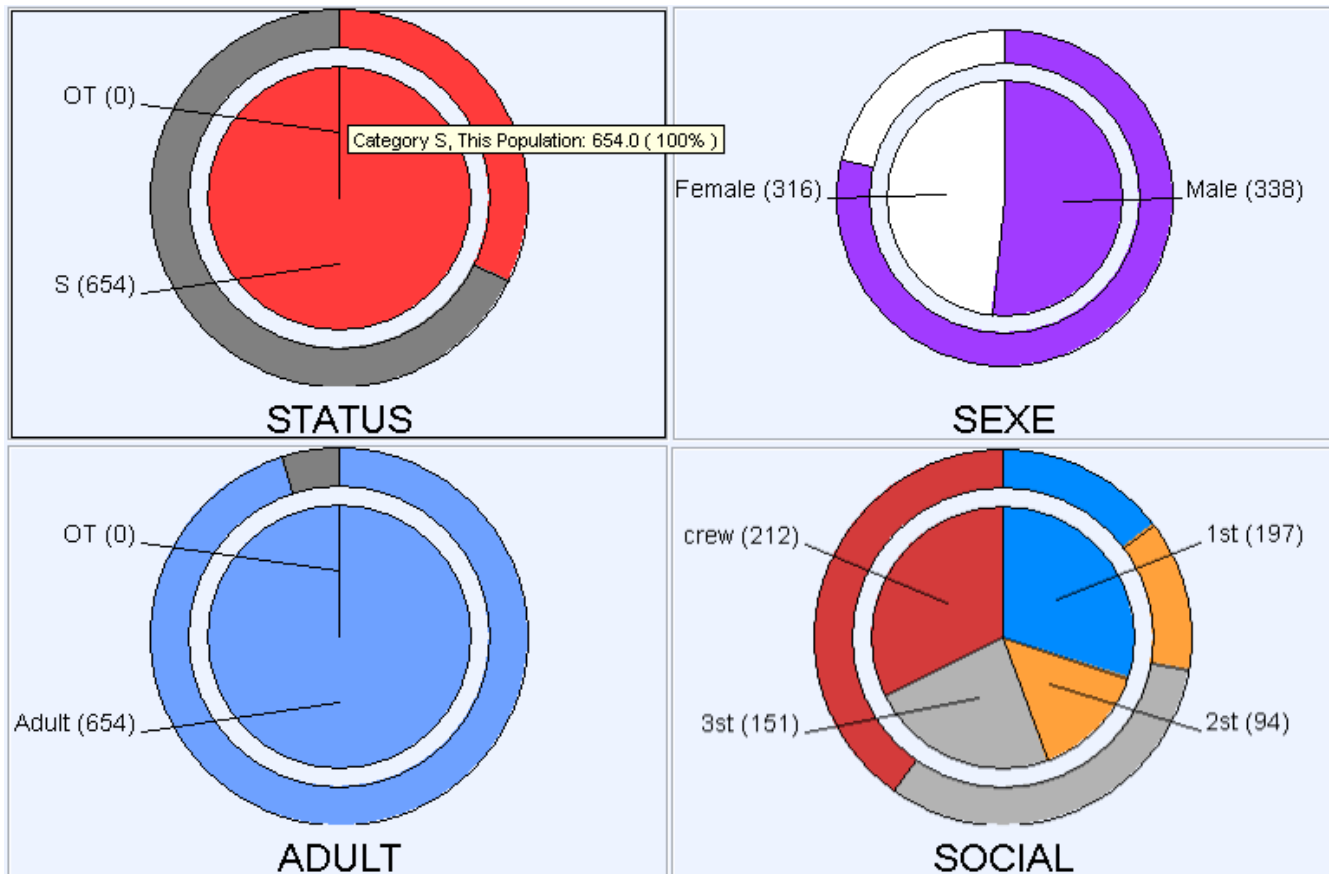


Illustration 44: Second Cluster found on the titanic data set.

This cluster contains 654 on the 2201 persons on board. All the persons were adult and survived.

The first interesting thing is that this cluster contains an higher percentage of female than the whole population which is more masculine. This cluster contains more than 68 % of the women on board and these 68 % survived. This cluster has also 20% of the male on board which survived.

Thanks to the 2 previous clusters, we could say that the rule “children and men before” is respected. But a closer look to the social category shows a worst information. The money has also a lot of influence.

The third cluster contains 66,7 % of the population. It is composed of in most cases with adult, men that were working in the crew. To conclude, I would like to modify the rule to “Children and women before the men and ordered by the social class”.

11 *PRESENTATION OF MY PROJECT IN GREECE*

In the end of my training period, I had the opportunity to present my work to teachers and administrators of the ARISTOTLE University of Thessaloniki. I had two meetings with the team over there. The first time to install the DB2 UDB on the system of the university and then to present to the Business Intelligence. The presentation lasted five hours.

The installation in the university was problematic due to proxy and firewall problems but we successful resolved them. One server has been installed, another one has been updated to DB2 V9.1.1 and some clients has been installed also.

12 ENCOUNTERED AND RESOLVED PROBLEMS

Here is a list of the occurred errors and encountered problems:

- Installing DB2 Data Warehouse in English on a Microsoft Windows XP French Version.
 - DB2 DWE is installed in the same language of the Operating System. The only way to be able to choose it is too have Microsoft Windows XP in MultiLanguale.
- The installation manual provided by I.B.M is outdated. The PART numbers are wrong and the client is not installed successfully if some modifications.
- Firewall problems (the port 5000 must be opened and the servers must accept ICMP Ping in order to activate the discovery mode).

13 CONCLUSION

We are already in the end of this report. The aim of my project was too try out the new I.B.M DB2 Data Warehouse Enterprise Edition. The Data warehouse has been build thanks to a Data Warehousing which runs control flows that are sequenced data flows.

The next part was to use to Intelligent Miner to build Mining Model. The models are clustering on the titanic using geometric and Kohonen algorithms, association rules and classification.

Both Data Warehouse and Data mining goals were successfully reached. The presented tutorials at the ARISTOTLE University, were a proof of the success of the project.

This training period was for me an unforgettable experience where I have learned to work without continuous assistance. I have been more than initiated to the data mining and that was the most of important part of my training period. From the beginning, I expect to learn about Knowledge Discovery In Database. The data mining is a fantastic tool but need a very good comprehension of the need of a company before integrating it in their system. I have been working for a fictional company but in the future I hope to continue working in this range of data mining and KDD.

In the other hand, an Erasmus project is also a the best way to improve my English, learn more about foreign culture and also to move out of the “family cocoon”. I think that I have been responsible during these three months in Thessaloniki. I had my own apartment, my own kitchen and so on. I thing that more than the work aspect, this project learns me a lot of things about the life in general.

The personal and professional goals have both been reached and we can conclude that this project was a real contribution in term of technical knowledge and personal experience.

14 **BIBLIOGRAPHY**

Books:

Chuck Ballard, "Dimensional Modeling: In a Business Intelligence Environment", RedBook march 2006.

Daniel M. Farrell, "Leveraging DB2 Data Warehouse Edition for Business Intelligence", RedBook, November 2006.

" White papers":

Dervos Dimitris and Anita Coleman, "A Common Sense Approach to Defining Data, Information, and Metadata ", 2006.

Dervos Dimitris and Haaga Helia, "Knowledge discovery from databases and data mining", march 2007.

Website:

I.B.M. Information Center,
<http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp?topic=>

David Dowe's data links, Titanic data set source,
<http://www.csse.monash.edu.au/~dld/datalinks.html>

15 APPENDICES

15.1 The Java bean code to create the data mining model of titanic data set

```

public class jb_ClustTita {

    private Connection connection;

    private Statement stmt;

    private List sqlStatements;

    /**
     * Constructor
     */
    public jb_ClustTita() {

        sqlStatements = new Vector();
        StringBuffer sqlStmt = new StringBuffer();
        sqlStmt.append("SET CURRENT SCHEMA=\"TEST\" ");
        sqlStatements.add(sqlStmt.toString());

        // Loading the data
        // A view is created to make modification easier using a WHERE
condition
        // for example : restrict the data set to the adult and so on.
        sqlStmt = new StringBuffer();
        sqlStmt.append("CREATE VIEW DATATITANIC AS ");
        sqlStmt.append(" SELECT ");
        sqlStmt.append("     \"SOCIAL\" AS \"SOCIAL\", ");
        sqlStmt.append("     \"ADULT\" AS \"ADULT\", ");
        sqlStmt.append("     \"SEX\" AS \"SEX\", ");
        sqlStmt.append("     \"STATUS\" AS \"STATUS\" ");
        sqlStmt.append(" FROM \"TEST\".\"TITANIC\" ");
        sqlStatements.add(sqlStmt.toString());

        // Creation of the model calling the Easy Mining For BASIC data mining

        sqlStmt = new StringBuffer();
        sqlStmt.append("CALL
IDMMX.BuildClusModel('titamaining.IM_CLUSTER_TITA', ");
        sqlStmt.append(" 'DATATITANIC', ");
        sqlStmt.append(" 'DM_setAlgorithm('Demographic')') "); // Kohonen is
the other algorithm
        sqlStatements.add(sqlStmt.toString());
        // Fill the view with the mining results.

```

```

        sqlStmt = new StringBuffer();
        sqlStmt.append("CREATE VIEW $MODELPORT$CLUSTERING_05 ( ");
        sqlStmt.append("  $LOC$MODEL, ");
        sqlStmt.append("  $LOC$TABLENAME, ");
        sqlStmt.append("  $LOC$MODELCOLNAME, ");
        sqlStmt.append("  $LOC$IDCOLNAME, ");
        sqlStmt.append("  $LOC$ID) AS ");
        sqlStmt.append("  (SELECT MODEL, 'IDMMX.CLUSTERMODELS', 'MODEL',
'MODELNAME', MODELNAME ");
        sqlStmt.append("    FROM IDMMX.CLUSTERMODELS ");
        sqlStmt.append("    WHERE MODELNAME = 'titamainig.IM_CLUSTER_TITA'
");

        sqlStatements.add(sqlStmt.toString());

        sqlStmt = new StringBuffer();
        sqlStmt.append("DROP VIEW $MODELPORT$CLUSTERING_05 ");
        sqlStatements.add(sqlStmt.toString());

        sqlStmt = new StringBuffer();
        sqlStmt.append("DROP VIEW DATATITANIC ");
        sqlStatements.add(sqlStmt.toString());

    }

    /**
     * Executes the Mining operation defined in this bean.
     * @param jndiName JNDI name of the data source for the operation.
     * @throws SQLException if a database access error occurred.
     * @throws NamingException if a data source lookup error occurred.
     */
    public void execute(String jndiName) throws SQLException, NamingException {
        Context context = new InitialContext();
        DataSource dataSource = (DataSource) context.lookup(jndiName);
        this.connection = dataSource.getConnection();
        connection.setAutoCommit(false);
        this.stmt = connection.createStatement();
        //run SQL statements
        Iterator it = sqlStatements.iterator();
        for (int i = 0; it.hasNext(); i++) {
            String sqlStmt = (String) it.next();
            this.stmt.execute(sqlStmt);
        }
        this.stmt.close();
    }
}

```