|  |
| --- |
| , RD Dep. |
| MTN.\*NIX.07 Oracle DB. Introduction to DWH |
| MTX.\*NIX.07. Introduction to DWH - Exit Task |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status of document | [**Kiryl Bucha**](mailto:Kiryl_bucha@epam.com) | 02-JAN-2012 |  |  |
|  |  |  |  |  |  |

*Contents*

[1. Exit Task 3](#_Toc320774863)

[2. Overview 3](#_Toc320774864)

[2.1. Business Background 3](#_Toc320774865)

[2.2. Benefits 3](#_Toc320774866)

[3. Requirements 3](#_Toc320774867)

[3.1. Business Requirements 3](#_Toc320774868)

[3.2. Technical Requirements 3](#_Toc320774869)

[4. Solution Sketch 3](#_Toc320774870)

[4.1. Source Tables structure 3](#_Toc320774871)

[4.2. Summarize Data Plan 4](#_Toc320774872)

[5. DWH Solution Concept 4](#_Toc320774873)

[5.1. Logical Diagram 4](#_Toc320774874)

[5.2. Physical diagram 4](#_Toc320774875)

[5.3. Dimensions 4](#_Toc320774876)

[5.3.1. Dimension Types 4](#_Toc320774877)

[5.3.2. Dimension Hierarchies 4](#_Toc320774878)

[5.4. Facts 4](#_Toc320774879)

[5.4.1. Facts Aggregations 4](#_Toc320774880)

[5.5. Dataflow Diagram 4](#_Toc320774881)

[5.6. Partitioning rules 4](#_Toc320774882)

[5.7. Strategy of Parallel execution 4](#_Toc320774883)

# Exit Task

Create next separate documents:

* BI Solution Proposal

**Business Background**

Business is a big chain of stores, witch sales products of apple(mobile phones, computers) and software for the products. Each purchase marks in order. If we want to have a well going business, we need to do the audit and collect statistic to have possibility see the trends, piks and causes that they have caused.

BI will help us to collect, transform, keep and analis information about products, customers, stores, orders.

* BI Solution Concept

We a going to develop BI solution to collect, transform, keep and analis information about products, customers, stores, orders by aggregating info about sales. Each sale is marked in order, so we need to work with orders.

The main requirement would to collect all elements that let us to build agrigation data about orders.

They are:

* Products attributes:

- type;

- sub type

- name,

- code,

- type configuration,

- list of technical specifications(display, size and weigt, storage, memory and so on),

- the year of appear,

- price,

- amount of products in store,

- no qualitative,

- country of produce,

- Warranty,

- gists,

"PROD\_DESC

"PROD\_WEIGHT\_CLASS"

"PROD\_UNIT\_OF\_MEASURE"

"PROD\_PACK\_SIZE"

"PROD\_STATUS"

"PROD\_VALID"

- Customer attributes:

"CUST\_FIRST\_NAME"

"CUST\_LAST\_NAME

"CUST\_GENDER"

"CUST\_YEAR\_OF\_BIRTH"

"CUST\_MARITAL\_STATUS

"CUST\_STREET\_ADDRESS"

"CUST\_POSTAL\_CODE"

"CUST\_CITY"

"CUST\_STATE\_PROVINCE"

"CUST\_MAIN\_PHONE\_NUMBER"

"CUST\_INCOME\_LEVEL"

"CUST\_CREDIT\_LIMIT"

"CUST\_EMAIL"

"CUST\_TOTAL"

"CUST\_EFF\_FROM",

"CUST\_EFF\_TO"

"CUST\_VALID"

* Stores:

Code,

Name,

Location,

Manager,

Date of first open,

Work time of store,

Holidays, weekends,

Rent

And so on.

# Overview

## Business Background

Our Business is a big chain of stores, witch sales products of apple(mobile phones, computers) and software for the products. Each purchase marks in order. If we want to have a well going business, we need to do the audit and collect static to have possibility see the trends, piks and causes that they have caused.

## Benefits

BI will help us to collect, transform, keep and analis information about products, customers, stores, orders.

# Requirements

## Business Requirements

Business want to know aggregation info about sales. Each sale marks in order, so we need to work with orders. Business need info about count of orders, there sum. Also important information about customers who make orders, there geo locations, gender, age and so on. Very useful would be to know about store’s sales. Ability to view the consistent of order, would give very useful info about well sales good and not very popular, we would be able to we aggregate info in different depends. So, as usual answer – Our customers want to know all possible info about sales in different slices.

The main requirement would to collect all elements that let us to build agrigation data about orders.

They are:

* Products attributes:

- type;

- sub type

- name,

- code,

- type configuration,

- list of technical specifications(display, size and weigt, storage, memory and so on),

- the year of appear,

- price,

- amount of products in store,

- no qualitative,

- country of produce,

- Warranty,

- gists,

"PROD\_DESC

"PROD\_WEIGHT\_CLASS"

"PROD\_UNIT\_OF\_MEASURE"

"PROD\_PACK\_SIZE"

"PROD\_STATUS"

"PROD\_VALID"

- Customer attributes:

"CUST\_FIRST\_NAME"

"CUST\_LAST\_NAME

"CUST\_GENDER"

"CUST\_YEAR\_OF\_BIRTH"

"CUST\_MARITAL\_STATUS

"CUST\_STREET\_ADDRESS"

"CUST\_POSTAL\_CODE"

"CUST\_CITY"

"CUST\_STATE\_PROVINCE"

"CUST\_MAIN\_PHONE\_NUMBER"

"CUST\_INCOME\_LEVEL"

"CUST\_CREDIT\_LIMIT"

"CUST\_EMAIL"

"CUST\_TOTAL"

"CUST\_EFF\_FROM",

"CUST\_EFF\_TO"

"CUST\_VALID"

* Stores:

Code,

Name,

Location,

Manager,

Date of first open,

Work time of store,

Holidays, weekends,

Rent

And so on.

## Technical Requirements

Max time for opening report -15s;

Keep info from the beginning of business,

Work time – 24/7/365,

Hardware Requirements.

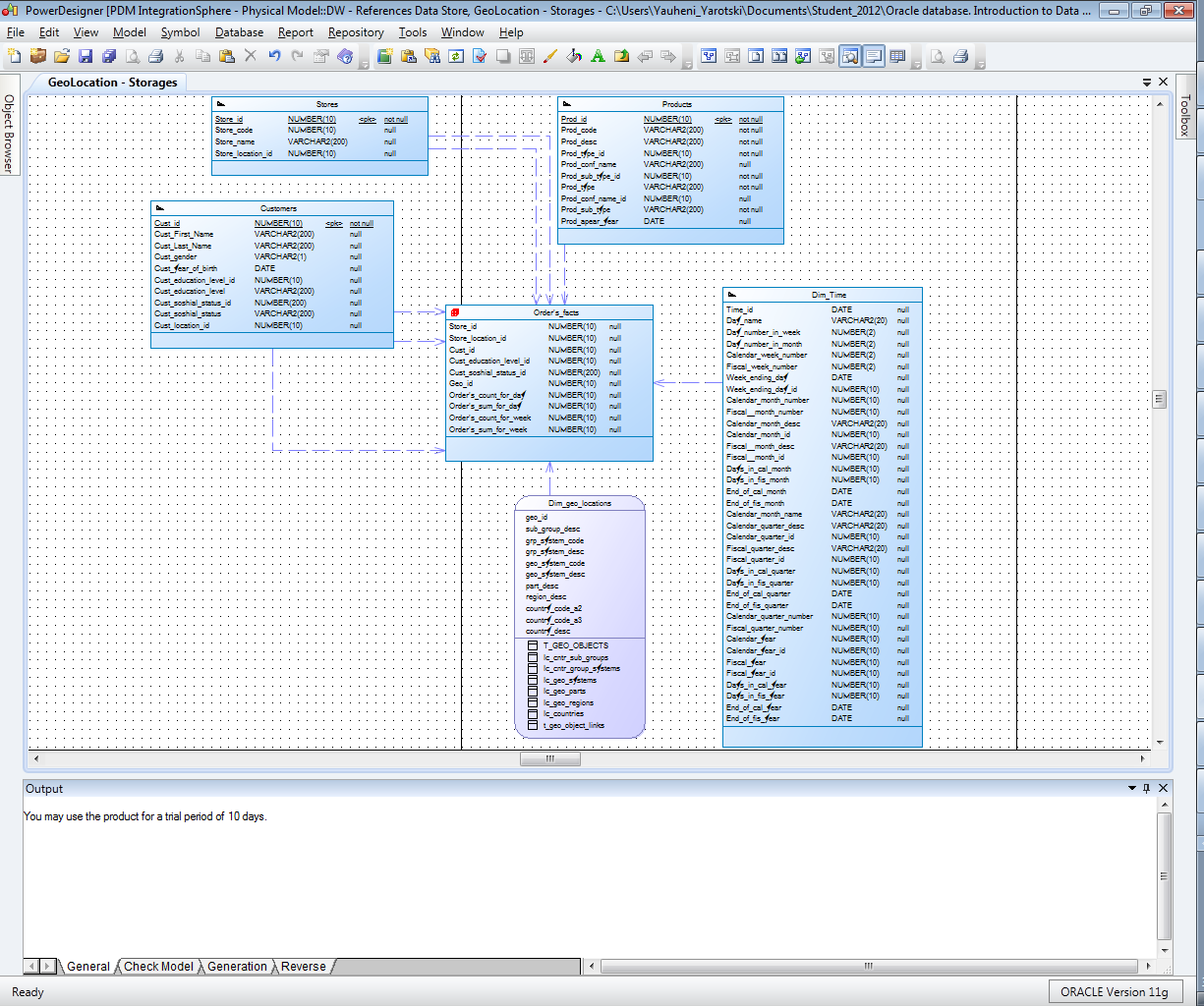
# Solution Sketch

## Source Tables structure

As u can see we have a lot of elements, but we don’t need them all. We have to choose only what we really need.

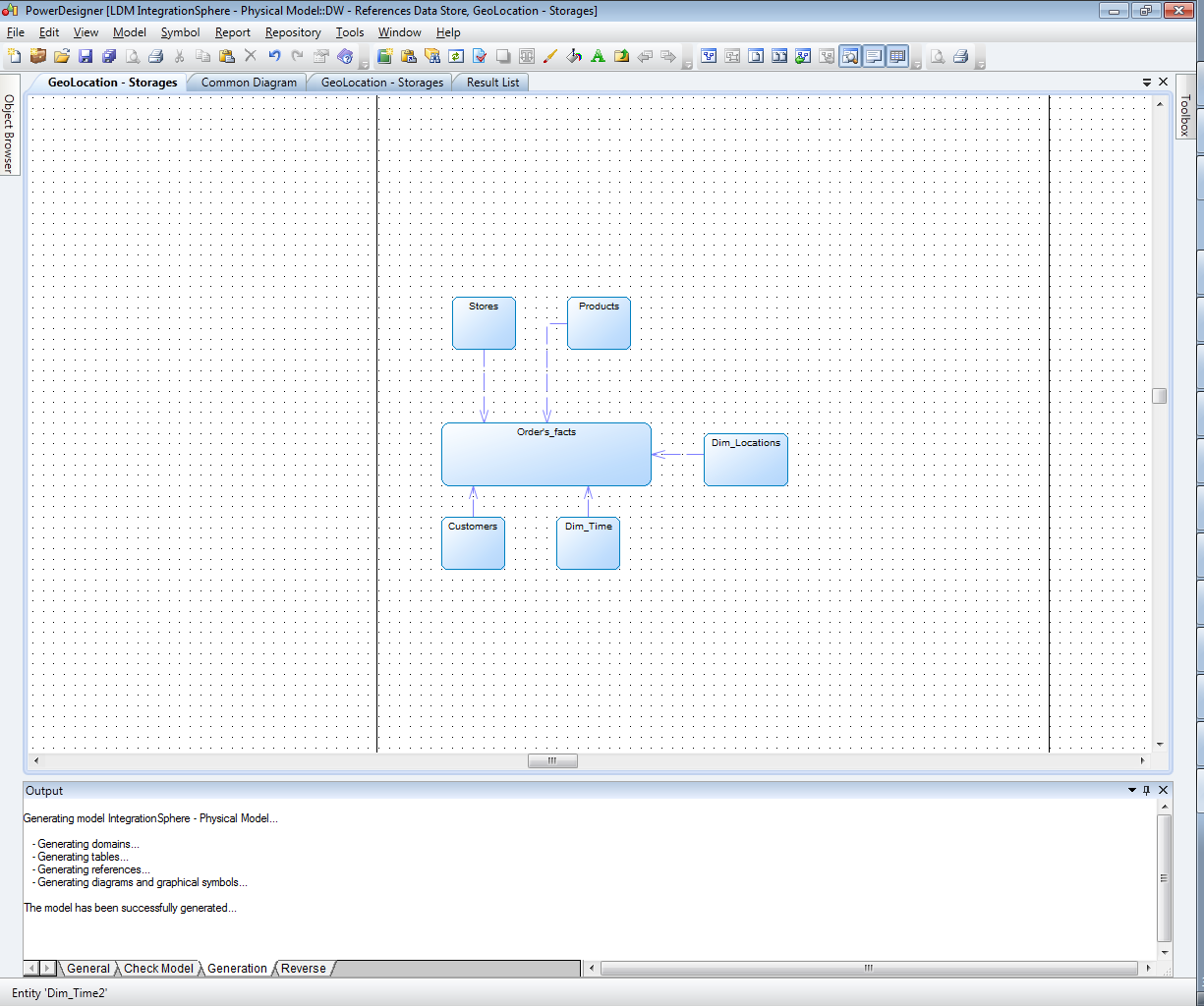
## Summarize Data Plan

Summarize data plan is shown on the diagram

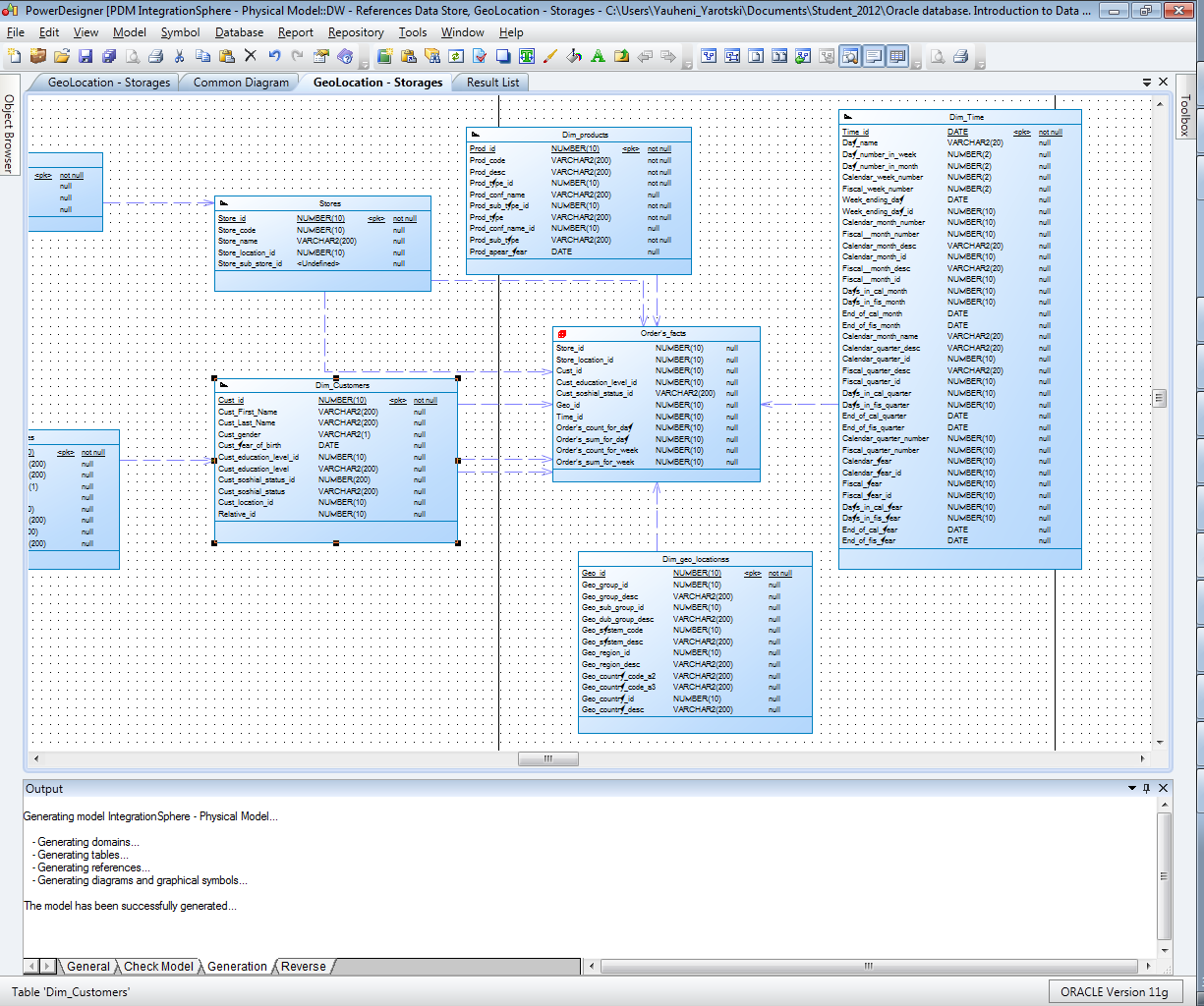


# DWH Solution Concept

## Logical Diagram



## Physical diagram



## Dimensions

### Dimension Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Size | DW – Merged Dimensions | Descriptions |
| DIM\_GEN\_TIME | SCD1 | BIG | DW.T\_DAYS, DW.T\_WEEKS, DW.T\_MONTHS, DW.T\_QUARTERS,  DW.T\_YEARS | TBD – Example row |
| Dim\_products | SCD2 | BIG | Dw.products,  Dw.prod\_type,  Dw.prod\_sub\_type,  Dw.prod\_conf\_type | In Dim\_products we have field “Prod\_apear\_year”. It means that each year we will have new dimension record. The surrogate key will change, prod name – won’t and Prod\_apear\_year – will. |
| DIM\_TIME | SCD2 | BIG | Dw\_days,  Dw\_weeks,  Dw\_months,  Dw\_quarters,  Dw\_yaers | Dim consists days, weeks, months, quarters and so on. |
| DIM\_GEO\_LOCATIONS | SCD3 | BIG | lc\_cntr\_group\_systems, lc\_cntr\_groups,  lc\_cntr\_sub\_groups,  t\_geo\_object\_links,  lc\_countries,  lc\_geo\_regions,  lc\_geo\_parts,  lc\_geo\_systems,  t\_geo\_types | SDC3 because country may change name, but we also have to be able to work with old and new names. |
| DIM\_GEN\_PERIODS | **CD** | Small |  |  |

### Dimension Hierarchies

**Hierarchy DAY-WEEK-MONTH-YEAR**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | LEVEL\_CODE | LEVEL\_DESC | LEVEL\_NATURAL\_KEY |
| DAYs | DAY | Store all day at the calendar | DAY\_ID |
| WEEKs | WEEK | Store all weeks at the calendar year | WEEK\_ID |
| **MONTHs** | **MONTH** | Store all **MONTHs** at the calendar year | **MONTH\_id** |
| **YEARs** | **YEAR** | Store all **YEARs** at the business calendar | **YEAR\_id** |

**Hierarchy DAY--MONTH- QUARTER -YEAR**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | LEVEL\_CODE | LEVEL\_DESC | LEVEL\_NATURAL\_KEY |
| DAYs | DAY | Store all day at the calendar | DAY\_ID |
| MONTHSs | MONTH | Store all months at the calendar year | WEEK\_ID |
| **QUARTERs** | **QUARTER** | Store all **QUARTER**s at the calendar year | **QUARTER\_id** |
| **YEARs** | **YEAR** | Store all **YEARs** at the business calendar | **YEAR\_id** |

**DIM\_products**

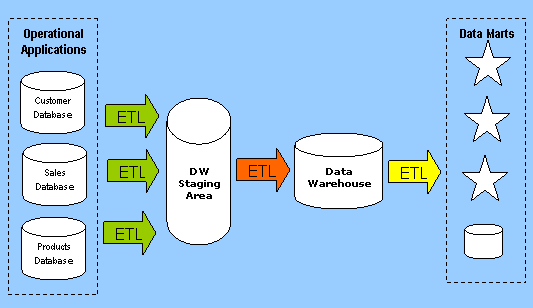
|  |  |  |  |
| --- | --- | --- | --- |
| Name | LEVEL\_CODE | LEVEL\_DESC | LEVEL\_NATURAL\_KEY |
| Prod\_types | Prod\_type | Store all types of products | Prod\_type\_id |
| Prod\_sub\_types | Prod\_sub\_type | Store all sub\_types of products | Prod\_sub\_type\_id |
| Prod\_names | Prod\_name | Store all Prod\_names of products list | Prod\_id |

## Facts

### Facts Aggregations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Code | Table Name | Additive | Descriptions |
| Counts Contracts | count\_cntr | FCT\_TEST | + | Calculate distinct values of Contracts at the event\_dt period. |
| Counts Order's, Order’s Sum | Store\_id, Store\_location\_id,  Cust\_id,  Cust\_education\_level\_id,  Cust\_soshial\_status\_id,  Geo\_id,  Order's\_count\_for\_day,  Order's\_sum\_for\_day | Store, Customers,  Geo\_locations, | + | Calculate distinct values and sum of orders by deminsions. |

## Dataflow Diagram



## Partitioning rules

create table "u\_dw\_references"."Order's\_facts"

(

"Store\_id" NUMBER(10),

"Store\_location\_id" NUMBER(10),

"Cust\_id" NUMBER(10),

"Cust\_education\_level\_id" NUMBER(10),

"Cust\_education\_level3" VARCHAR2(200),

"Geo\_id" NUMBER(10),

"Time\_id" NUMBER(10),

"Order's\_count\_for\_day" NUMBER(10),

"Order's\_sum\_for\_day" NUMBER(10),

"Order's\_count\_for\_week" NUMBER(10),

"Order's\_sum\_for\_week" NUMBER(10)

)

partition by range

("Time\_id")

interval

((numtodsinterval(1,'day')))

(

PARTITION part\_1 VALUES LESS THAN (to\_date('01/01/2010','dd/mm/yyyy')),

PARTITION part\_2 VALUES LESS THAN (to\_date('01/01/2011','dd/mm/yyyy'))

)

store in ("xz\_ts\_sata\_01")

(partition compress);

## Strategy of Parallel execution