**Types of Facts in Data Warehouse**

A fact table is the one which consists of the measurements, metrics or facts of business process. These measurable facts are used to know the business value and to forecast the future business. The different types of facts are explained in detail below.  
  
**Additive:**  
  
Additive facts are facts that can be summed up through all of the dimensions in the fact table. A sales fact is a good example for additive fact.  
  
**Semi-Additive:**   
  
Semi-additive facts are facts that can be summed up for some of the dimensions in the fact table, but not the others.  
Eg: Daily balances fact can be summed up through the customers dimension but not through the time dimension.  
  
**Non-Additive:**   
  
Non-additive facts are facts that cannot be summed up for any of the dimensions present in the fact table.   
Eg: Facts which have percentages, ratios calculated.

**Factless Fact Table:**

In the real world, it is possible to have a fact table that contains no measures or facts. These tables are called "Factless Fact tables".   
  
Eg: A fact table which has only product key and date key is a factless fact. There are no measures in this table. But still you can get the number products sold over a period of time.

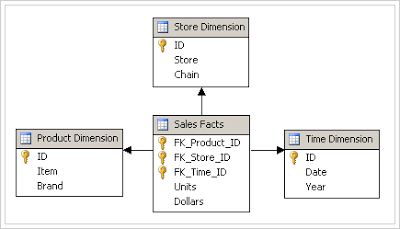
A fact tables that contain aggregated facts are often called summary tables.

**Types of Dimensions in data warehouse**

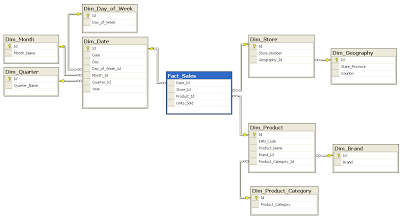
A dimension table consists of the attributes about the facts. Dimensions store the textual descriptions of the business. With out the dimensions, we cannot measure the facts. The different types of dimension tables are explained in detail below.  
  
**Conformed Dimension:**  
  
Conformed dimensions mean the exact same thing with every possible fact table to which they are joined.   
  
Eg: The date dimension table connected to the sales facts is identical to the date dimension connected to the inventory facts.  
  
**Junk Dimension:**  
  
A junk dimension is a collection of random transactional codes flags and/or text attributes that are unrelated to any particular dimension. The junk dimension is simply a structure that provides a convenient place to store the junk attributes.  
  
Eg: Assume that we have a gender dimension and marital status dimension. In the fact table we need to maintain two keys referring to these dimensions. Instead of that create a junk dimension which has all the combinations of gender and marital status (cross join gender and marital status table and create a junk table). Now we can maintain only one key in the fact table.  
  
**Degenerated Dimension:**  
  
A degenerate dimension is a dimension which is derived from the fact table and doesn't have its own dimension table.  
  
Eg: A transactional code in a fact table.  
  
**Role-playing dimension:**  
  
Dimensions which are often used for multiple purposes within the same database are called role-playing dimensions. For example, a date dimension can be used for “date of sale", as well as "date of delivery", or "date of hire".

**Data Warehouse Dimensional Modelling (Types of Schemas)**

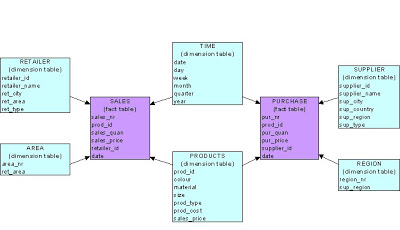
There are four types of schemas are available in data warehouse. Out of which the star schema is mostly used in the data warehouse designs. The second mostly used data warehouse schema is snow flake schema. We will see about these schemas in detail.  
  
**Star Schema:**  
  
A star schema is the one in which a central fact table is sourrounded by denormalized dimensional tables. A star schema can be simple or complex. A simple star schema consists of one fact table where as a complex star schema have more than one fact table.

[](http://3.bp.blogspot.com/_pjSOGJIjDMo/S1w8XBuw8jI/AAAAAAAAADI/8SMt2Io0P-A/s1600-h/star-schema.png)

**Snow Flake Schema:**  
  
A snow flake schema is an enhancement of star schema by adding additional dimensions. Snow flake schema are useful when there are low cardinality attributes in the dimensions.

[](http://1.bp.blogspot.com/_pjSOGJIjDMo/S1w_JRMx3XI/AAAAAAAAADQ/J6cKyxJiwbM/s1600-h/Snowflake-schema.png)

**Galaxy Schema:**  
  
Galaxy schema contains many fact tables with some common dimensions (conformed dimensions). This schema is a combination of many data marts.

[](http://1.bp.blogspot.com/_pjSOGJIjDMo/S1w_SAaqJBI/AAAAAAAAADY/_kYiR3xwbCQ/s1600-h/galaxy.bmp)

**Fact Constellation Schema:**  
  
The dimensions in this schema are segregated into independent dimensions based on the levels of hierarchy. For example, if geography has five levels of hierarchy like teritary, region, country, state and city; constellation schema would have five dimensions instead of one.

**SCD type 4 - Fast growing dimension**

In normal scenarios the dimension tables tend to grow slowly. That is the reason, they are called slowly changing dimensions. Example: Location attribute of a customer changes very rarely. However the salary band of a customer is likely to change every year. These type of attributes causes the customer dimension table to grow rapidly.   
  
SCD type 4 provides a solution to handle the rapid changes in the dimension tables. The concept lies in creating a junk dimension or a small dimension table with all the possible values of the rapid growing attributes of the dimension.   
  
**Example**: Take a look at the following dimension attributes of customer

C\_Id

Name

Location

Age\_band

Salary\_band

The age band and salary band are going to change frequently. So create a separate small dimension table with only these attributes. The number of possible values of age band will be around 20 and salary band will be around 30. The total number of rows in the new dimension table are 20x30=60.   
  
The new tables are 

Table name : customer

C\_Id

Name

Location

Table name: customer\_mini

M\_id

Age\_band

Salary\_band

Fact table:

Id

C\_Id

M\_Id

----

The dimension key of the new table should be maintained in the fact table. This way we can handle the rapid changes in the dimension table.   
  
Data Warehouse Design Approaches

Data warehouse design is one of the key technique in building the data warehouse. Choosing a right data warehouse design can save the project time and cost. Basically there are two data warehouse design approaches are popular.  
  
**Bottom-Up Design:**  
  
In the bottom-up design approach, the data marts are created first to provide reporting capability. A data mart addresses a single business area such as sales, Finance etc. These data marts are then integrated to build a complete data warehouse.  The integration of data marts is implemented using data warehouse bus architecture. In the bus architecture, a dimension is shared between facts in two or more data marts. These dimensions are called conformed dimensions. These conformed dimensions are integrated from data marts and then data warehouse is built.  
  
**Advantages of bottom-up design are:**

* This model contains consistent data marts and these data marts can be delivered quickly.
* As the data marts are created first, reports can be generated quickly.
* The data warehouse can be extended easily to accommodate new business units. It is just creating new data marts and then integrating with other data marts.

**Disadvantages of bottom-up design are:**

* The positions of the data warehouse and the data marts are reversed in the bottom-up approach design.

**Top-Down Design:**  
  
In the top-down design approach the, data warehouse is built first. The data marts are then created from the data warehouse.  
  
**Advantages of top-down design are:**

* Provides consistent dimensional views of data across data marts, as all data marts are loaded from the data warehouse.
* This approach is robust against business changes. Creating a new data mart from the data warehouse is very easy.

**Disadvantages of top-down design are:**

* This methodology is inflexible to changing departmental needs during implementation phase.
* It represents a very large project and the cost of implementing the project is significant.

# Extraction Methods in Data Warehouse

The extraction methods in data warehouse depend on the source system, performance and business requirements. There are two types of extractions, Logical and Physical. We will see in detail about the logical and physical designs.  
  
**Logical extraction**  
  
There are two types of logical extraction methods:  
  
**Full Extraction**: Full extraction is used when the data needs to be extracted and loaded for the first time.  In full extraction, the data from the source is extracted completely. This extraction reflects the current data available in the source system.  
  
**Incremental Extraction**: In incremental extraction, the changes in source data need to be tracked since the last successful extraction. Only these changes in data will be extracted and then loaded. These changes can be detected from the source data which have the last changed timestamp. Also a change table can be created in the source system, which keeps track of the changes in the source data.  
  
One more method to get the incremental changes is to extract the complete source data and then do a difference (minus operation) between the current extraction and last extraction. This approach causes a performance issue.  
  
**Physical extraction**  
  
The data can be extracted physically by two methods:  
  
**Online Extraction**:  In online extraction the data is extracted directly from the source system. The extraction process connects to the source system and extracts the source data.  
  
**Offline Extraction**: The data from the source system is dumped outside of the source system into a flat file. This flat file is used to extract the data. The flat file can be created by a routine process daily.

## **What are the Dimensions in Data warehouse?**

A dimension table consists of the attributes about the facts. Dimensions store the textual descriptions of the business attribute. Without the dimensions, we cannot measure the facts and facts are just disordered Numbers.In Business, Customer, Products, Buyers information can be different dimensions.

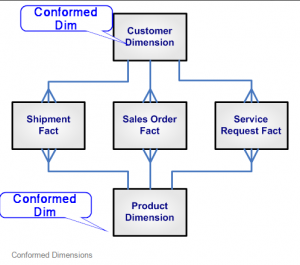
Let’s walk through commonly used types of dimensions.

## **Types of commonly used Dimensions in data warehouse Design**

* Conformed Dimensions
* Junk Dimensions
* Role-playing Dimensions
* Slowly Changing Dimensions
* Degenerated Dimensions

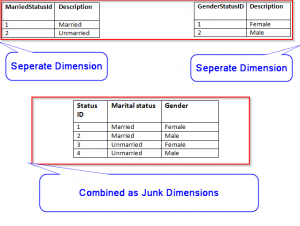
##### **Conformed Dimensions**

A Dimension that is used in multiple locations is called conformed dimensions. A conformed dimension may be used with multiple fact tables in single database, or across multiple data marts or Data warehouses.

[](http://sqlserver-qa.net/wp-content/uploads/2015/06/Conformed_dimensions.png)

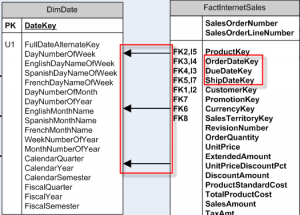
I.e. Above shown Customer and Product Dimensions are Conformed Dimensions as they are connected to Shipment Fact table, Sales Order Fact table, and Service Request Fact table.

**Junk Dimensions**

A junk dimension is a collection of random transaction codes flags and/or text attributes that are unrelated to any particular dimension. The junk dimension is simply a structure that provides a convenient place to store the junk attributes.  
[](http://sqlserver-qa.net/wp-content/uploads/2015/06/JunkDimension.png)  
**I.e.**:  Assume that we have a gender dimension and marital status dimension. In the fact table we need to maintain two keys referring to these dimensions. Instead of that create a junk dimension which has all the combinations of gender and marital status (cross join gender and marital status table and create a junk table). Now we can maintain only one key in the fact table.

**Role-playing Dimensions**

Role Playing Dimensions are the Dimensions which often used for multiple purposes within same database.Here same dimension key is associated with more than one foreign key in the fact table in the database for the different purposes.

[](http://sqlserver-qa.net/wp-content/uploads/2015/06/RolePlayingDim.png)

**I.e.**:  In Date dimensions, [**FullDateAlternateKey]** is associated with [**Orderdate key], [Duedate key], and [Shipdate]**key in the fact table to solve different purpose in Data warehouse.

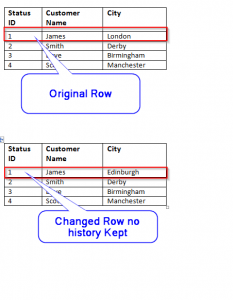
##### **Slowly Changing Dimensions**

This is widely used Dimensions type. It is the dimensions where attribute values changes with time. There are various types of Slowly Changing Dimensions (SCD) based on how business manages this dimensions.

**Types of SCD**

TYPE 0: It is the dimensions where we do not change attribute values at all. They are rarely used. **I.e.** Employee birth date

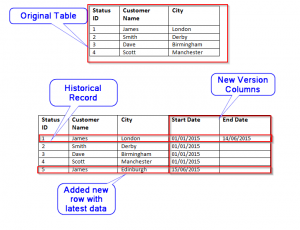
TYPE 1: In this type, Old value of attribute is overwritten by new values of attribute and no history kept

[](http://sqlserver-qa.net/wp-content/uploads/2015/06/scd_1.png)

**I.e Customer City** where company decided to show only current one.

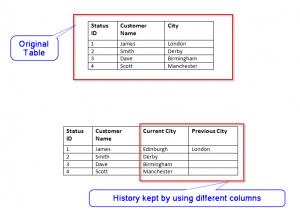
In this case previous city name London is replaced by new city name Edinburgh.

TYPE 2: In this type we tracks historical data by creating multiple records for a given Natural key (business key) in the dimensional tables with separate surrogate key and/or different version numbers. Unlimited history is preserved for each insert.

[](http://sqlserver-qa.net/wp-content/uploads/2015/06/scd2.png)

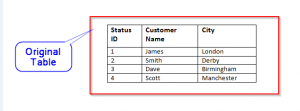
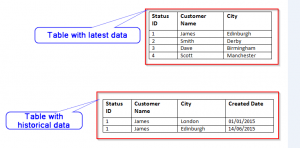
**I.e.** **Customer City** where company decided to have historical data then we will have to add an extra row with column to identify the **Current/Historical** attributes value by start and end date columns.

TYPE 3: In this type, we tracks changes using separate columns and preserves limited history.it is limited to how many columns we want to add in dimension table.

[](http://sqlserver-qa.net/wp-content/uploads/2015/06/scd3.png)

**I.e.** Customer City where New columns “previous City” and “Current City” being added.

TYPE 4: In this type, we keep all or some historical data in separate table and current data stays in main Dimension table. Both historical and current dimension table joined to fact table with same surrogate key, this will enhance the query performance. This type    used very rarely.

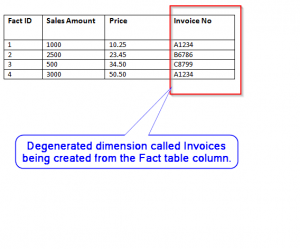
[](http://sqlserver-qa.net/wp-content/uploads/2015/06/scd_4_2.png) 

**I.e.** we create new table to store previous Customer City and Current Customer City in Historical table with Created date And Current Customer city in Current dimension table.

##### **Degenerated Dimensions**

A degenerate dimension is a dimension which is derived from the fact table and doesn’t have its own dimension table.

In Data warehouse this Dimension often used to show drill through capability where in the report you can see how aggregated no came up.

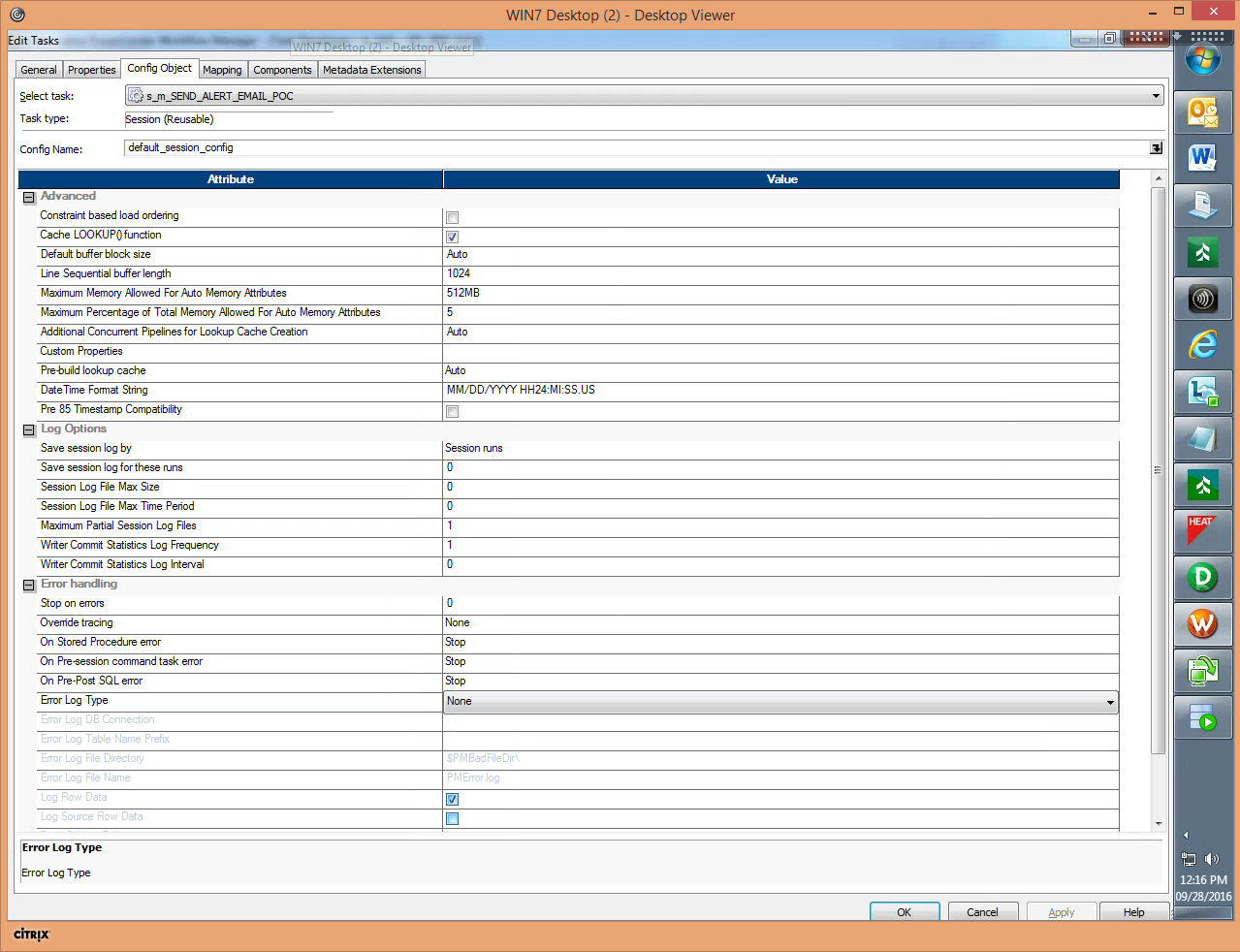


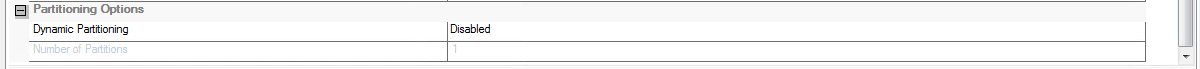
**I.e.** Invoice no can be stored in the fact table and then used as separate dimensions for the drill through purpose to find out what invoices are part of total buying cost in report.

So Dimensions are one of the pillar of the data warehouse.Choosing right one can define future of the data warehouse.It always good to use right type

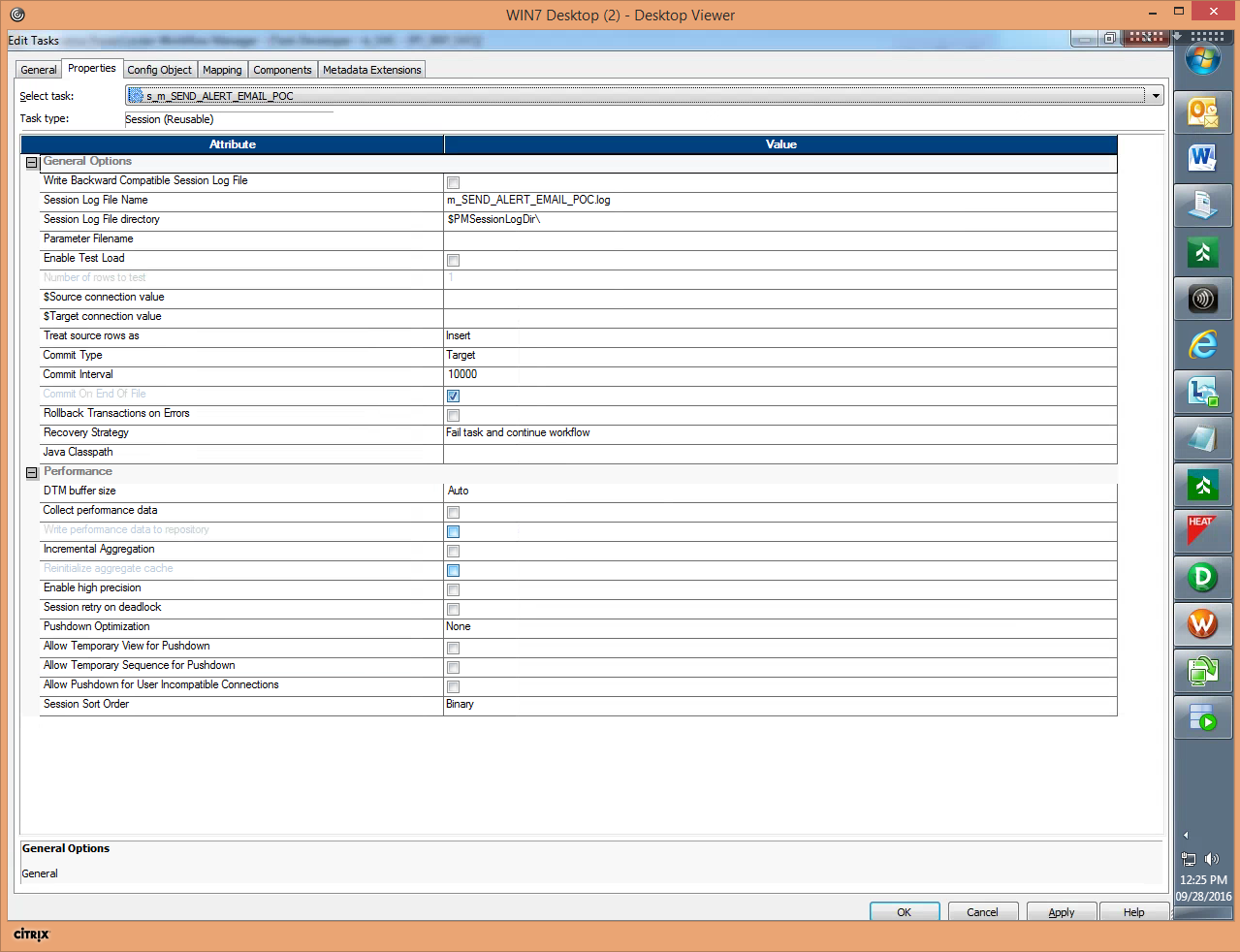
SESSION:

* 1. SESSION CONFIG OBJECT

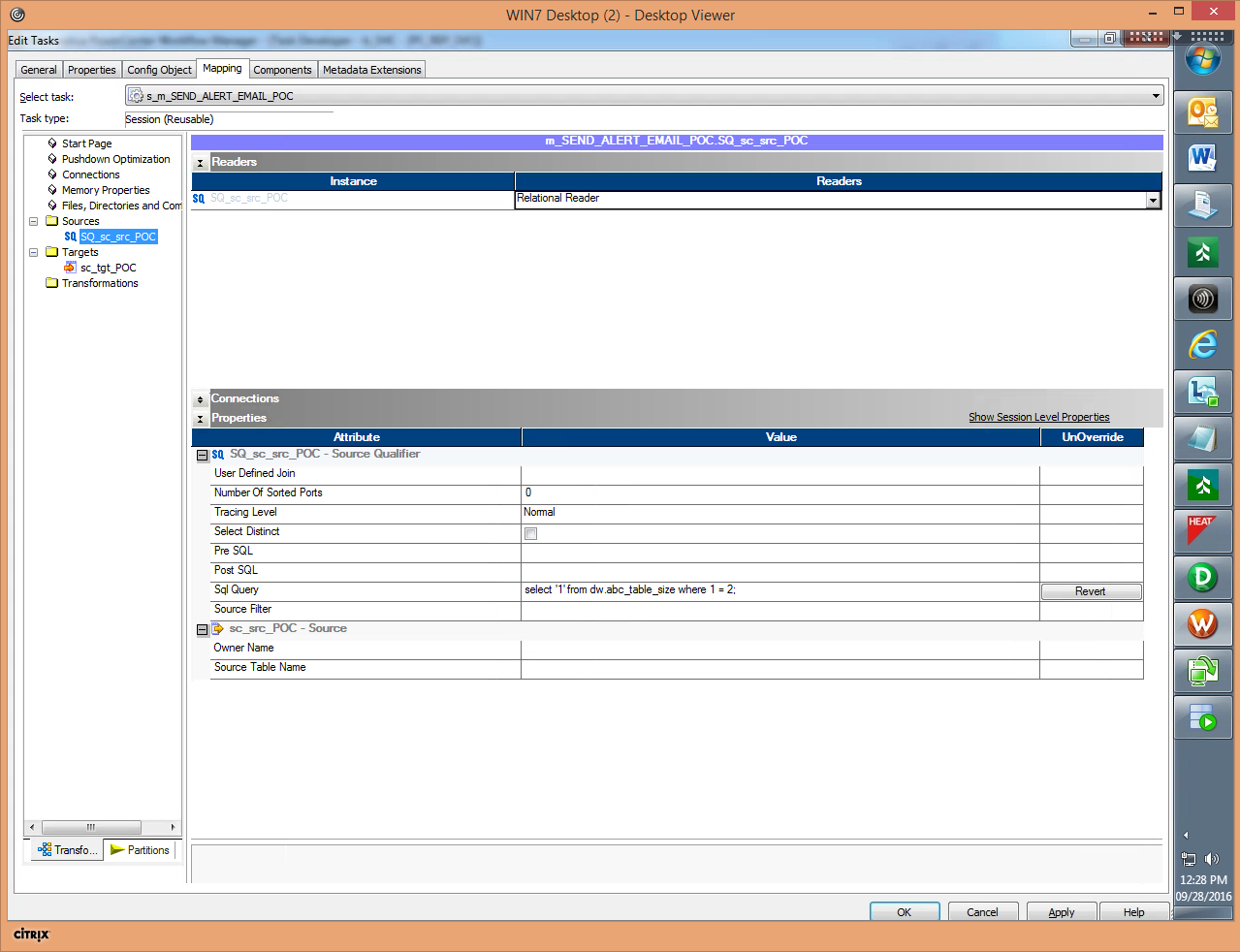


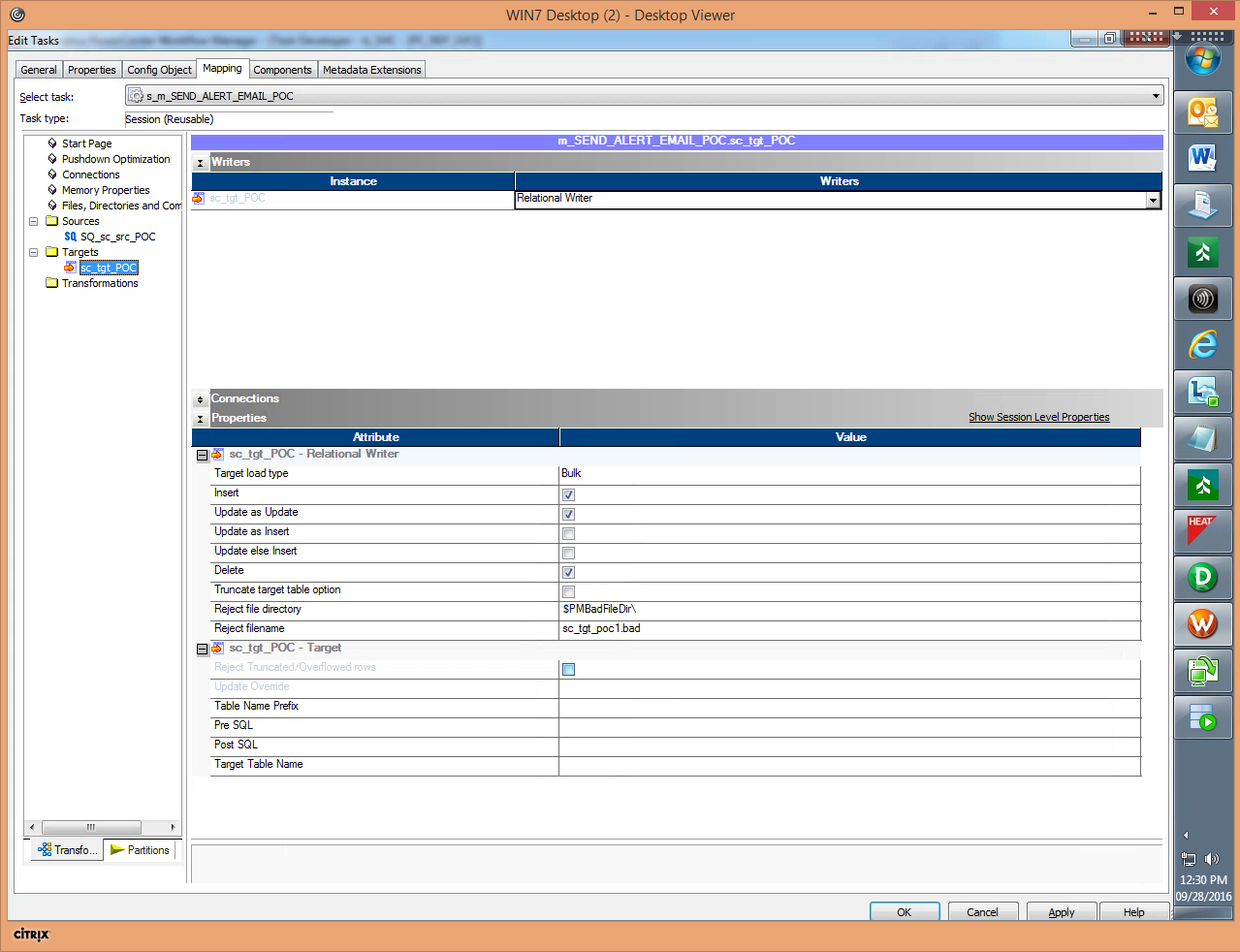


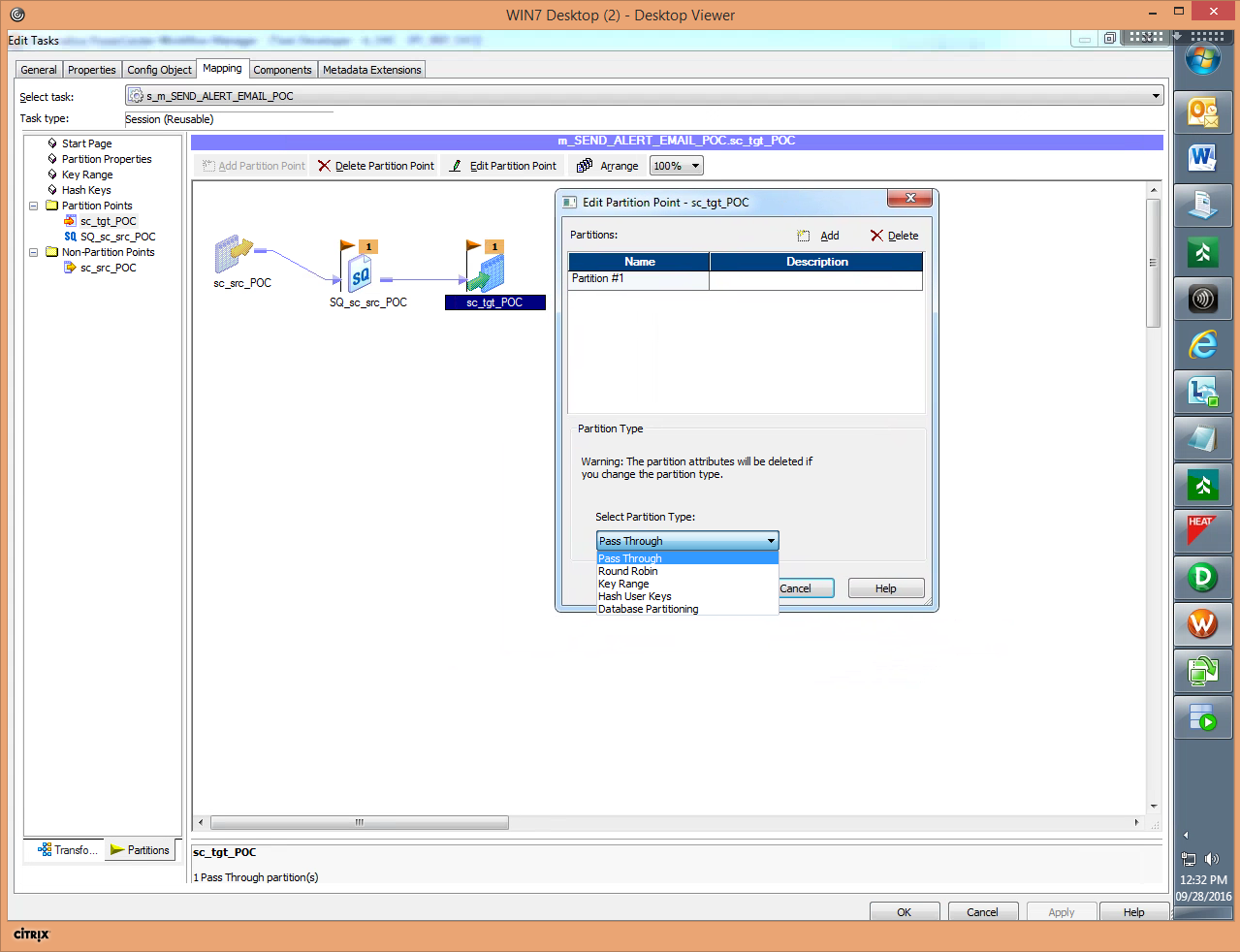
* 1. SESSION PROPERTIES:



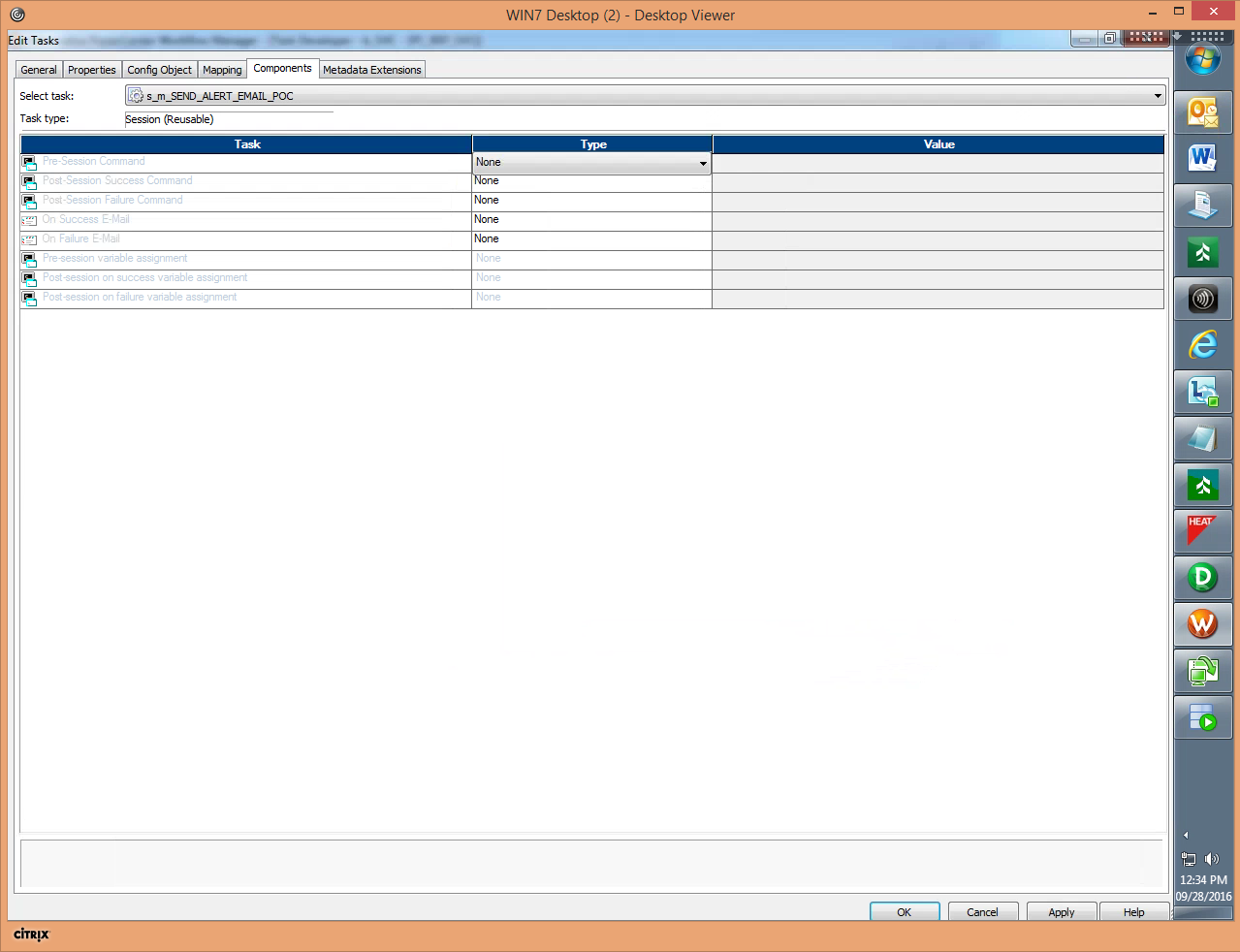
* 1. SESSION MAPPING



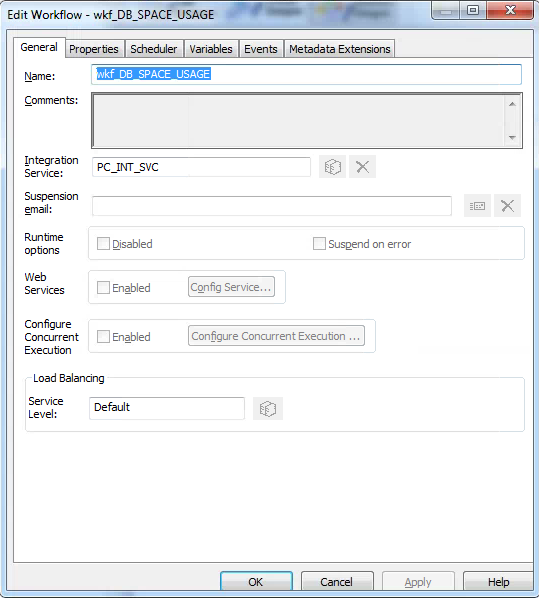


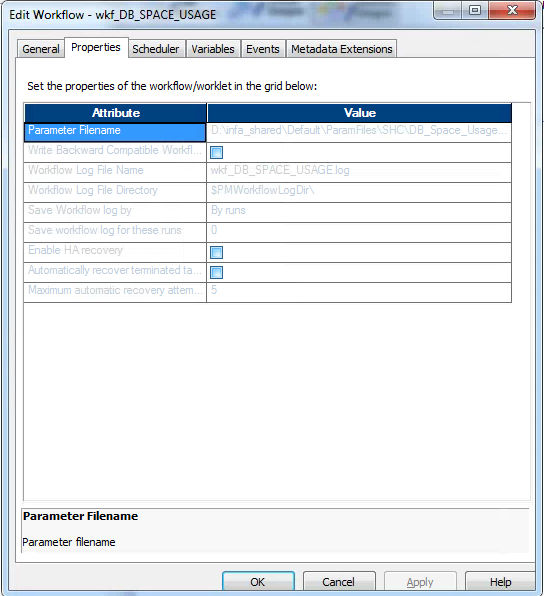


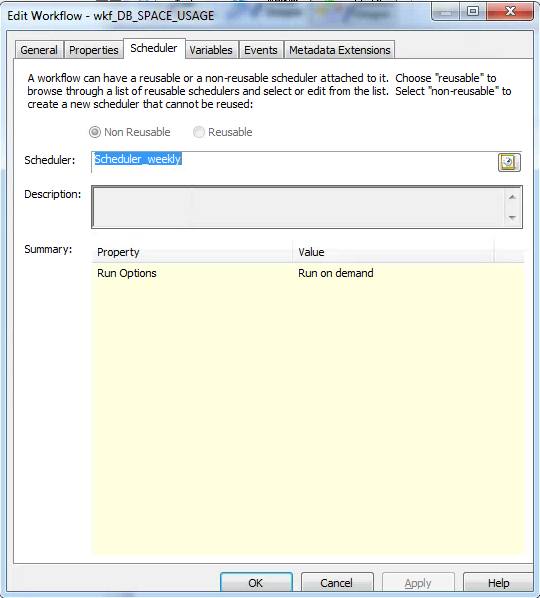
* 1. SESSION COMPONENTS:

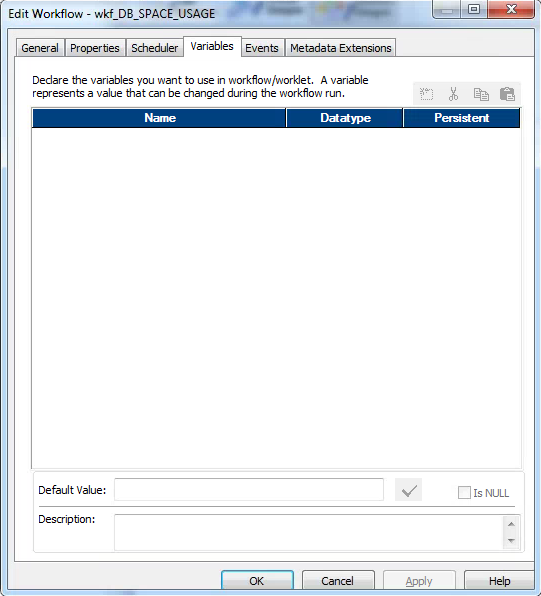


WORKFLOW:



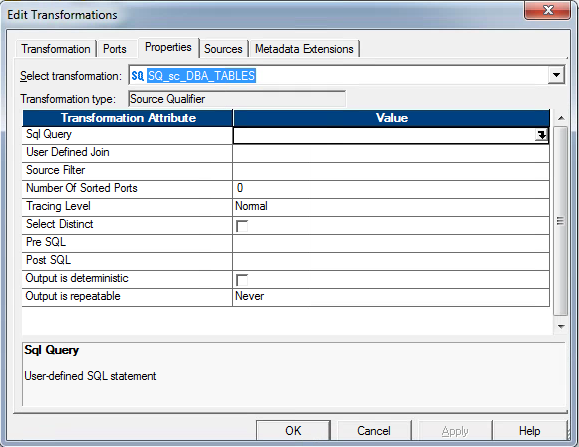




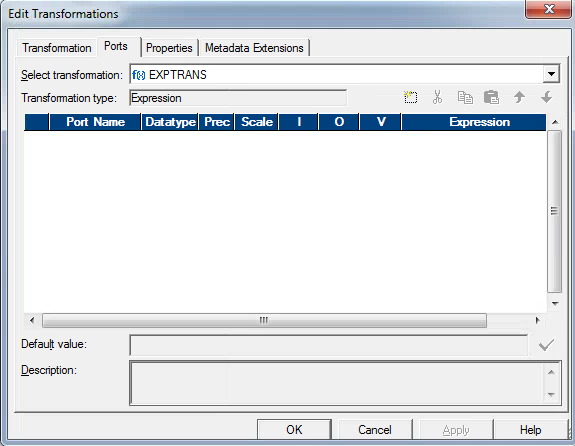


TRANSFORMATIONS:

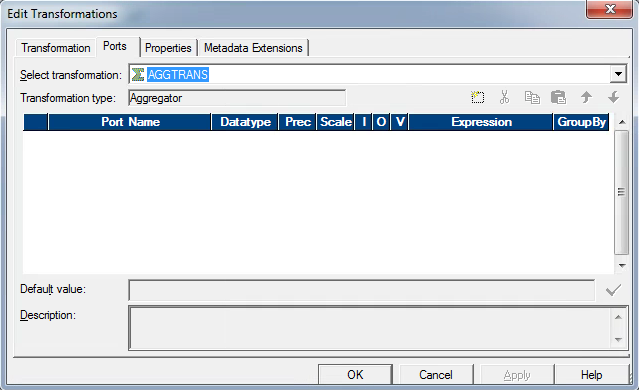
1. Source Qualifier:

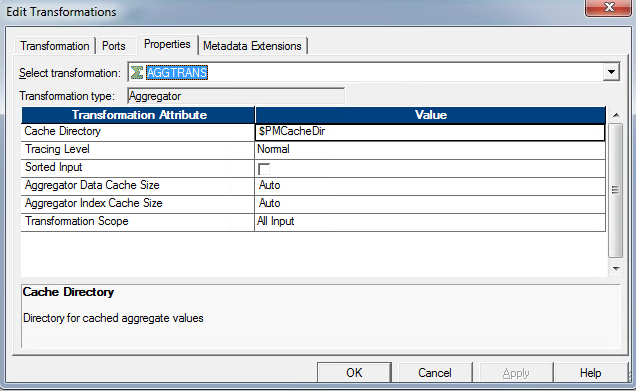


2. Expression:

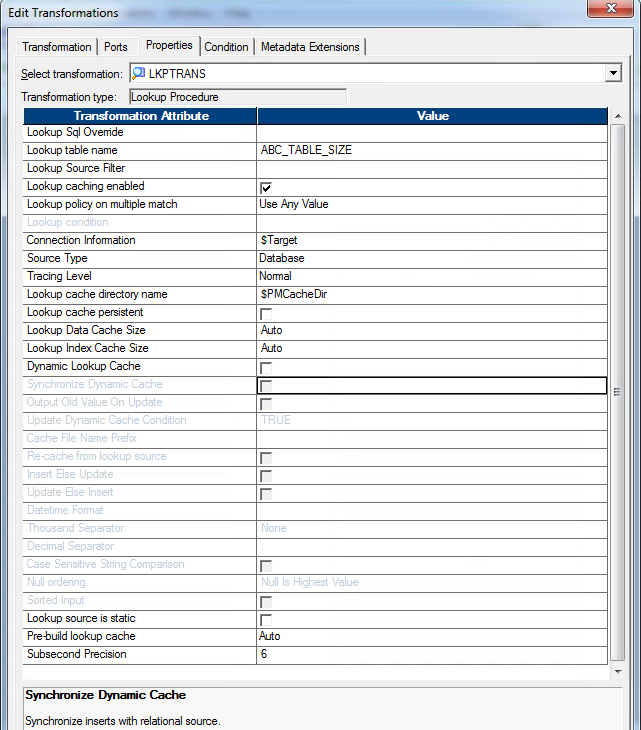


3. Aggregator

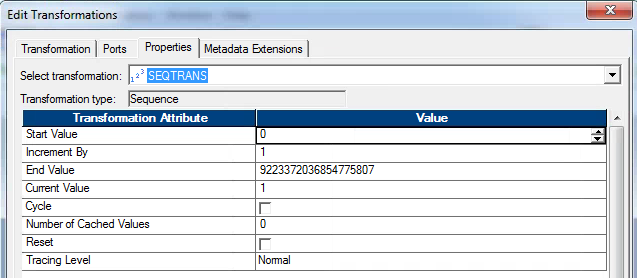




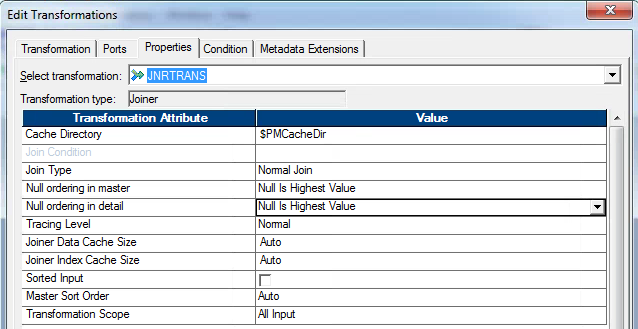
4. Lookup

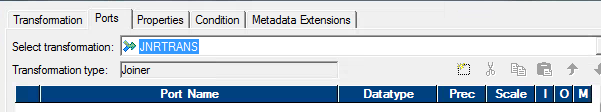


5. Sequence Generator

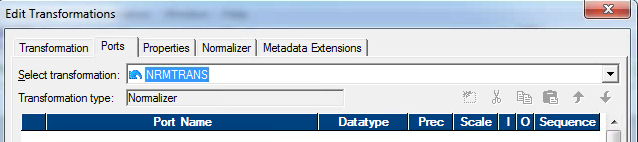


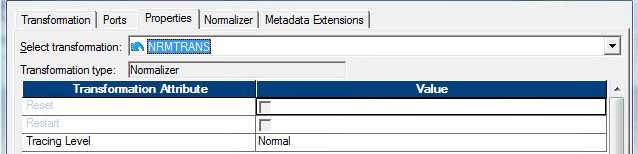
6. Joiner:

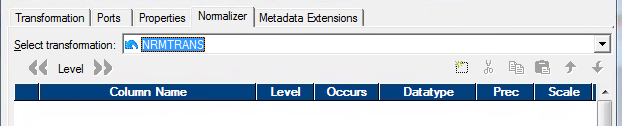




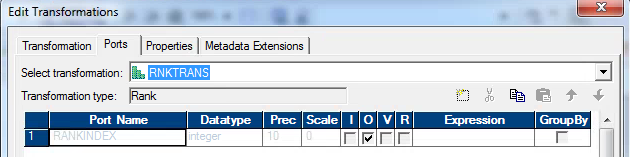
7. Normalizer:

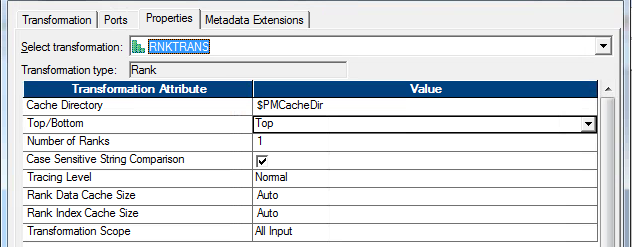




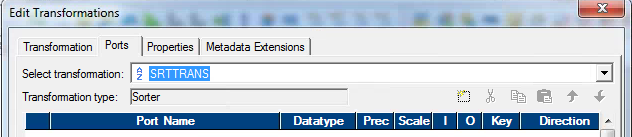


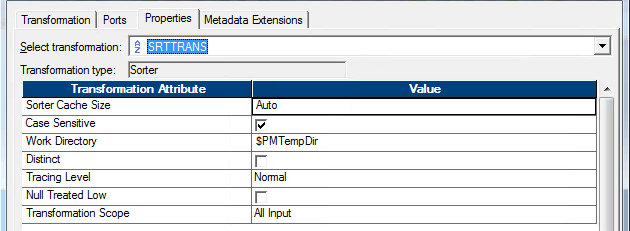
8. Rank:



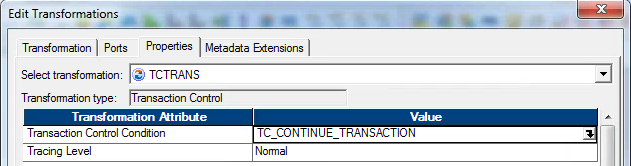


9. Sorter

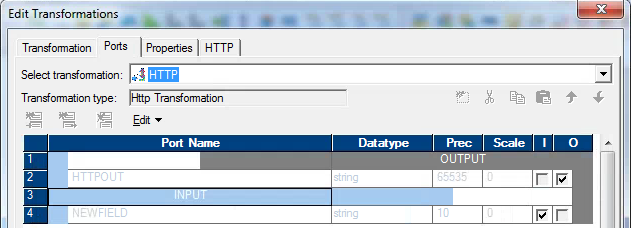


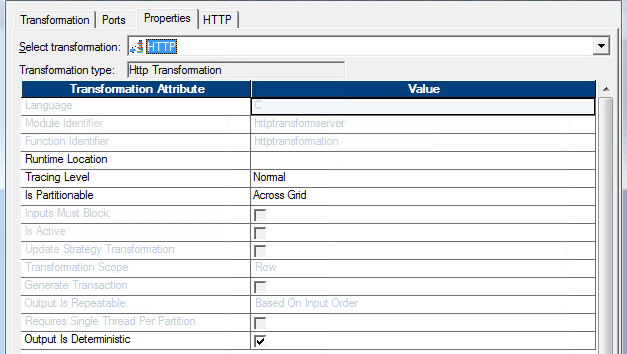


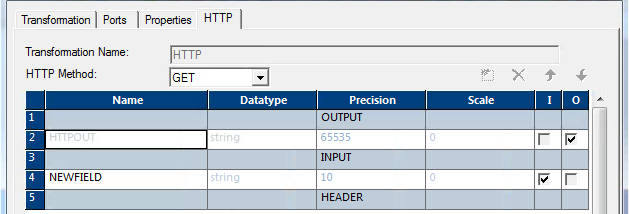
10. Transaction Control:



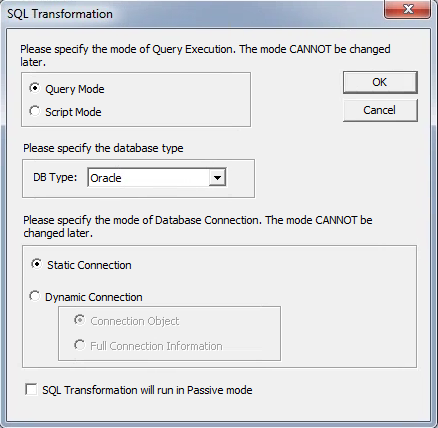
11. HTTP Transformation:

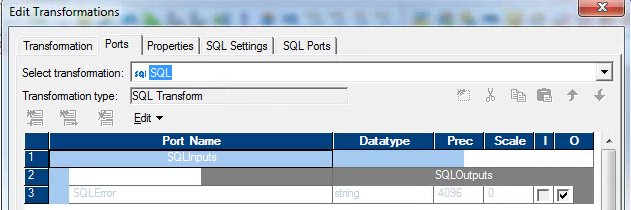


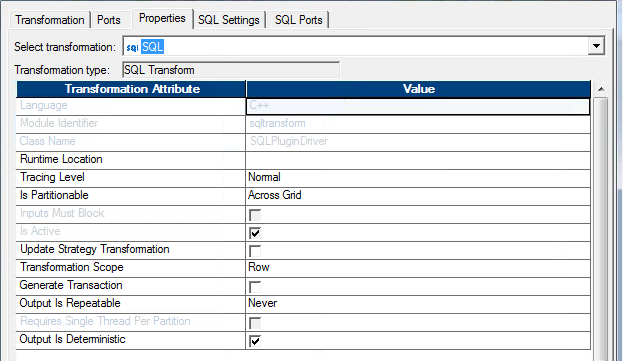


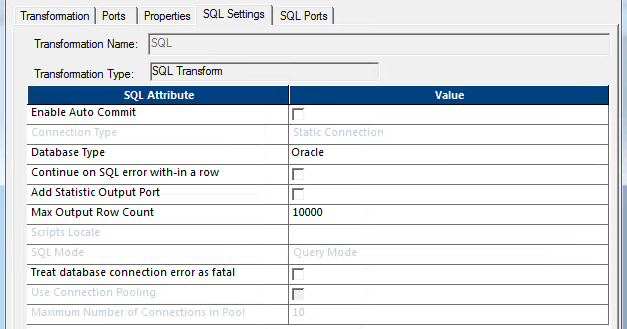


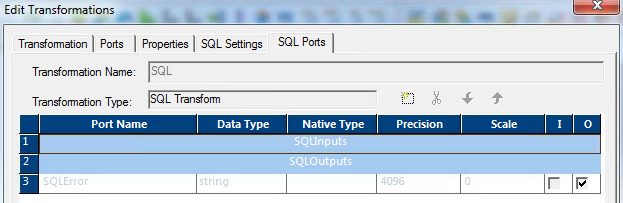
12. SQL transformation



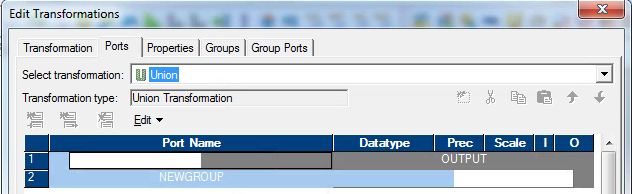


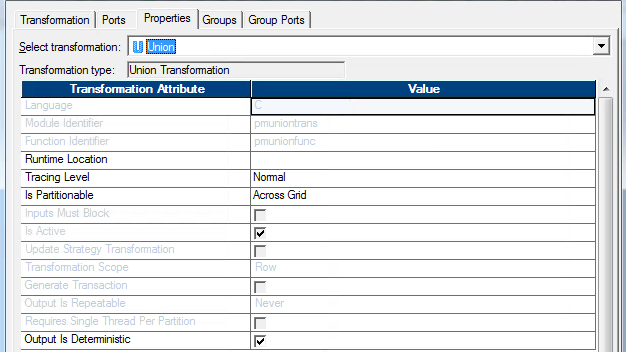


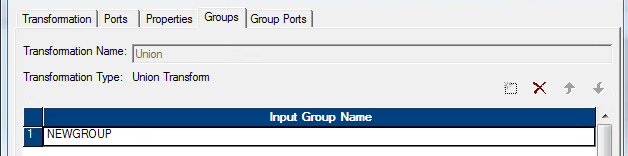




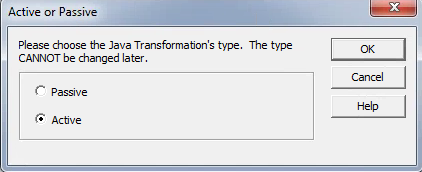
13. Union

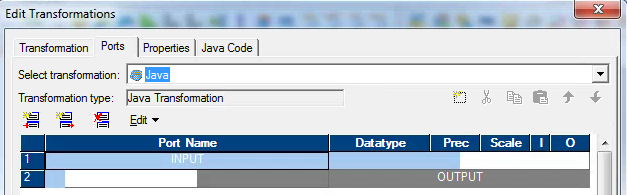


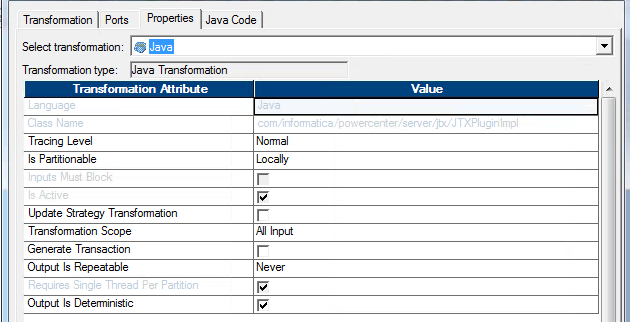


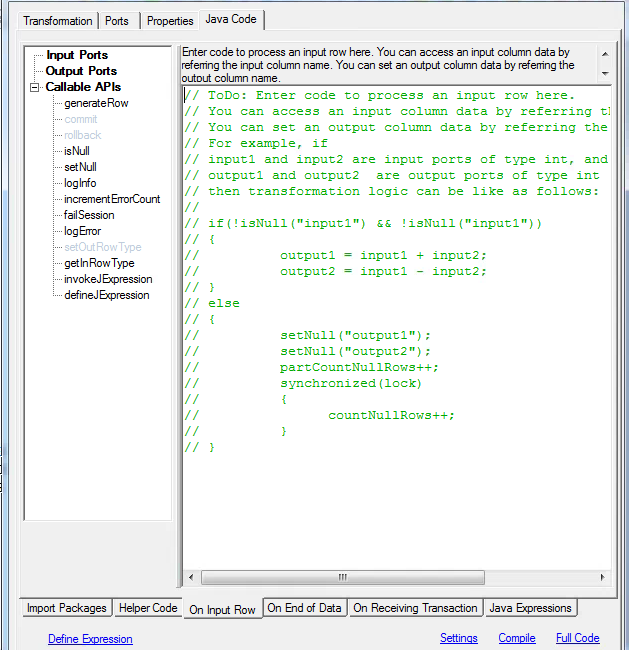


14. Java:









15. Stored Procedure Transformation

16. External Procedure Transformation

17. Custom Transformation

18. Web Service Consumer Transformation

19. XML Parser Transformation