[**Informatica**](http://vivek-informatica.blogspot.in/)

**Transformations Overview**

**Transformations Overview:**

A transformation is a repository object that generates, modifies, or passes data. The Designer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data. Transformations in a mapping represent the operations the Informatica Server performs on the data. Data passes into and out of transformations through ports that you connect in a mapping or mapplet.

Transformations can be active or passive. An active transformation can change the number of rows that pass through it, such as a Filter transformation that removes rows that do not meet the configured filter condition. A passive transformation does not change the number of rows that pass through it, such as an Expression transformation that performs a calculation on data and passes all rows through the transformation.

Transformations can be connected to the data flow, or they can be unconnected. An unconnected transformation is not connected to other transformations in the mapping. It is called within another transformation, and returns a value to that transformation.

|  |  |  |
| --- | --- | --- |
| Transformations and Descriptions | | |
| **Transformation** | **Type** | **Description** |
| Advanced External Procedure | Active/  Connected | Calls a procedure in a shared library or in the COM layer of Windows NT. |
| Aggregator | Active/  Connected | Performs aggregate calculations. |
| ERP Source Qualifier | Active/  Connected | Represents the rows that the Informatica Server reads from an ERP source when it runs a session. |
| Expression | Passive/  Connected | Calculates a value. |
| External Procedure | Passive/  Connected or Unconnected | Calls a procedure in a shared library or in the COM layer of Windows NT. |
| Filter | Active/  Connected | Filters records. |
| Input | Passive/  Connected | Defines mapplet input rows. Available only in the Mapplet Designer. |
| Joiner | Active/  Connected | Joins records from different databases or flat file systems. |
| Lookup | Passive/  Connected or Unconnected | Looks up values. |
| Normalizer | Active/  Connected | Normalizes records, including those read from COBOL sources. |
| Output | Passive/  Connected | Defines mapplet output rows. Available only in the Mapplet Designer. |
| Rank | Active/  Connected | Limits records to a top or bottom range. |
| Sequence Generator | Passive/  Connected | Generates primary keys. |
| Source Qualifier | Active/  Connected | Represents the rows that the Informatica Server reads from a relational or flat file source when it runs a session. |
| Router | Active/  Connected | Routes data into multiple transformations based on a group expression. |
| Stored Procedure | Passive/  Connected or Unconnected | Calls a stored procedure. |
| Update Strategy | Active/  Connected | Determines whether to insert, delete, update, or reject records. |
| XML Source Qualifier | Passive/  Connected | Represents the rows that the Informatica Server reads from an XML source when it runs a session. |

When you build a mapping, you add transformations and configure them to handle data according to your business purpose. Perform the following tasks to incorporate a transformation into a mapping:

* **Create the transformation.** Create it in the Mapping Designer as part of a mapping, in the Mapplet Designer as part of a mapplet, or in the Transformation Developer as a reusable transformation.
* **Configure the transformation.** Each type of transformation has a unique set of options that you can configure.
* **Connect the transformation to other transformations and target definitions.** Drag one port to another to connect them in the mapping or mapplet.

**Creating a Transformation:**

You can create transformations using the following Designer tools:

* **Mapping Designer.** Create transformations that connect sources to targets. Transformations in a mapping cannot be used in other mappings unless you configure them to be reusable.
* **Transformation Developer.** Create individual transformations, called reusable transformations that you can use in multiple mappings.
* **Mapplet Designer.** Create and configure a set of transformations, called mapplets, that you can use in multiple mappings.

Use the same process to create a transformation in the Mapping Designer, Transformation Developer, and Mapplet Designer.

**To create a transformation:**

* Open the appropriate Designer tool.
* In the Mapping Designer, open or create a Mapping. In the Mapplet Designer, open or create a Mapplet.
* On the Transformations toolbar, click the button corresponding to the transformation you want to create.

**OR**

* Choose Transformation-Create and select the type of transformation you want to create**.**
* Click and drag across the portion of the mapping where you want to place the transformation.
* The new transformation appears in the workspace. Next, you need to configure the transformation by adding any new ports to it and setting other properties.

**Creating Ports:**

You can create a new port in the following ways:

* **Drag a port from another transformation.** When you drag a port from another transformation the Designer creates a port with the same properties and connects the two ports. Choose Layout-Copy Columns to enable copying ports.
* **Click the Add button on the Ports tab.** The Designer creates an empty port you can configure.

**Configuring Ports:**

On the Ports tab, you can configure the following properties:

* **Port name.** The name of the port.
* **Datatype, Precision and Scale.** If you plan to enter an expression or condition, make sure the datatype matches the return value of the expression.
* **Port Type.** Transformations may contain a combination of Input, Output, Input/Output, and Variable port types.
* **Default Value.** The Designer assigns default values to handle null values and output transformation errors. You can override the default value in some ports.
* **Description.** A description of the port.
* **Other properties.** Some transformations have properties specific to that transformation, such as expressions or group by properties.

**Note:** The Designer creates some transformations with configured ports. For example, the Designer creates a Lookup transformation with output port for each column in the table or view used for the lookup. You need to create a port representing a value used to perform a lookup.

**Connecting Ports:**

Once you add and configure a transformation in a mapping, you connect it to other mapping objects. You connect mapping objects through the ports. Data passes into and out of a mapping through the following ports:

* **Input ports.** Receive data.
* **Output ports.** Pass data.
* **Input/output ports.** Receive data and pass it unchanged.

To connect ports, drag between ports in different mapping objects. The Designer validates the connection and creates the connection only when the connection meets validation requirements.

**Working with Expressions:**

You can enter expressions using the Expression Editor in some transformations. PowerMart and PowerCenter use a SQL-like language that contains many functions designed to handle common data transformations. For example, you can use the AVG function to find the average of all values in a column. Enter an expression in an output port that uses the value of data from an input or input/output port. For example, you have a transformation with an input port IN\_SALARY that contains the salaries of all your employees. You might want to use the individual values from the IN\_SALARY column later in the mapping, as well as the total and average salaries you calculate through this transformation. For this reason, the Designer requires you to create a separate output port for each calculated value.

|  |  |  |
| --- | --- | --- |
| List of Transformations Containing Expressions | | |
| **Transformation** | **Expression** | **Return Value** |
| Expression | Performs a calculation based on values within a single record. For example, based on the price and quantity of a particular item, you can calculate the total purchase price for that line item in an order. | The result of a row-level calculation for a port. |
| Aggregator | Performs an aggregate calculation based on all records passed through the transformation. Alternatively, you can specify a filter for records in the aggregate calculation to explicitly exclude certain kinds of records. For example, you can find the total number and average salary of all employees in a branch office using this transformation. | The result of an aggregate calculation for a port. |
| Filter | Specifies a condition used to filter records passed through this transformation. For example, if you want to write customer data to the BAD\_DEBT table for customers with outstanding balances, you could use the Filter transformation to filter customer records. | TRUE or FALSE, depending on whether a record meets the specified condition. Only records that return TRUE are passed through this transformation. The transformation applies this value to each record passed through it. |
| Update Strategy | Flags a record for update, insert, delete, or reject. You use this transformation when you want to exert fine control over updates to a target, based on some condition you apply. For example, you might use the Update Strategy transformation to flag all customer records for update when the mailing address has changed, or flag all employee records for reject for people no longer working for the company. | The numeric code for update, insert, delete, or reject. The transformation applies this value to each record passed through it. |
| Rank | Sets the conditions for records included in a rank. For example, you can rank the top 10 salespeople who are still employed with the company. | The result of a condition or calculation for a port. |
| Router | Routes data into multiple transformations based on a group expression. For example, you can use this transformation to compare the salaries of employees at three different pay levels. You can do this by creating three groups in the Router transformation. For example, create one group expression for each salary range. | TRUE or FALSE, depending on whether a row meets the specified group expression. Only rows that return TRUE pass through each user-defined group in this transformation. Rows that return FALSE automatically pass through the default group. |

**Using the Expression Editor:**

Use the point-and-click interface in the Expression Editor to minimize errors when you enter expressions. You can select functions, ports, and operators to build SQL-like statements for the Informatica Server.

**Adding Comments:**

The Expression Editor also allows you to add comments using -- or // comment indicators. You can use comments to give descriptive information about the expression, or you can specify a valid URL to access business documentation about the expression.

**Validating Expressions:**

Once you enter the expression, you can validate it by clicking the Validate button. If you do not validate an expression, the Designer automatically validates it when you close the Expression Editor. You cannot run a session against a mapping with invalid expressions.

**Expression Editor Display**

The Expression Editor can display syntax expressions in different colors for better readability. If you have the latest Rich Edit control, riched20.dll, installed on your system, the Expression Editor displays expression functions in blue, comments in grey, and quoted strings in green. You can resize the Expression Editor by dragging the borders of the dialog box.

**Adding Expressions to an Output Port**

Use the following steps to add an expression to an output port. To add expressions:

* In the transformation, select the port and open the Expression Editor.
* Enter the expression. Use the Functions and Ports tabs and the operator keys.
* Add comments to the expression. Use comment indicators -- or //.
* Validate the expression. Use the Validate button to validate the expression.

**Using Local Variables:**

You can use local variables in Aggregator, Expression, and Rank transformations. You can reference variables in an expression or use them to temporarily store data. Variables are an easy way to improve session performance. You might use variables to perform the following tasks:

* Temporarily store data.
* Simplify complex expressions.
* Store values from prior rows.
* Capture multiple return values from a stored procedure.
* Compare values.
* Store the results of an unconnected Lookup transformation.

**Temporarily Store Data and Simplify Complex Expressions:**

Variables improve performance when you enter several related expressions in the same transformation. Rather than parsing and validating the same expression components each time, you can define these components as variables. For example, if an Aggregator transformation uses the same filter condition before calculating sums and averages, you can define this condition once as a variable, and then reuse the condition in both aggregate calculations. This also allows you to simplify complex expressions. If an Aggregator includes the same calculation in multiple expressions, you can improve session performance by creating a variable to store the results of the calculation. For example, you might create the following expressions to find both the average salary and the total salary using the same data:

AVG (SALARY, ((JOB\_STATUS = ‘Full-time’) AND (OFFICE\_ID = 1000)))

SUM (SALARY, ((JOB\_STATUS = ‘Full-time’) AND (OFFICE\_ID = 1000)))

Rather than entering the same arguments for both calculations, you might create a variable port for each condition in this calculation, and then modify the expression to use the variables.

|  |  |
| --- | --- |
| Variable Usage: | |
| **Port** | **Value** |
| V\_CONDITION1 | JOB\_STATUS = ‘Full-time’ |
| V\_CONDITION2 | OFFICE\_ID = 1000 |
| AVG\_SALARY | AVG(SALARY, (V\_CONDITION1 AND V\_CONDITION2 ) ) |
| SUM\_SALARY | SUM(SALARY, (V\_CONDITION1 AND V\_CONDITION2 ) ) |

**Guidelines for Configuring Variable Ports:**

Consider the following factors when you configure variable ports in a transformation:

* **Port order.** The Informatica Server evaluates ports by dependency. The order in which ports display in a transformation must match the order of evaluation: input ports, variable ports, output ports.
* **Datatype.** The datatype you choose for reflects the return value of the expression you enter.
* **Variable Initialization.** The Informatica Server sets initial values in variable ports. This allows you to create counters.

**Port Order:**

The Informatica Server evaluates ports in the following order:

* **Input ports.** The Informatica Server evaluates all input ports first since they do not depend on any other ports. Therefore, you can create input ports in any order. Since they do not reference other ports, the Informatica Server does not order input ports.
* **Variable ports.** Variable ports can reference input ports and variable ports, but not output ports. Because variable ports can reference input ports, the Informatica Server evaluates variable ports after input ports. Likewise, since variable can reference other variables, the display order for variable ports is the same as the order in which the Informatica Server evaluates each variable.
* For example, if you calculate the original value of a building and then adjust for depreciation, you might create the original value calculation as a variable port. This variable port needs to appear before the port that adjusts for depreciation.
* **Output ports.** Because output ports can reference input ports and variable ports, the Informatica Server evaluates output ports last. The display order for output ports does not matter since output ports cannot reference other output ports. Be sure output ports display at the bottom of the list of ports.

**Datatype:**

When you configure a port as a variable, you can enter any expression or condition in it. The data type you choose for this port reflects the return value of the expression you enter. If you specify a condition through the variable port, any numeric data type returns the values for TRUE (non-zero) and FALSE (zero).

**Variable Initialization:**

The Informatica Server does not set the initial value for variables to NULL. Instead, the Informatica Server uses the following guidelines to set initial values for variables:

* Zero for numeric ports
* Empty strings for string ports
* 01/01/1753 for Date/Time ports if PMServer 4.0 date handling compatibility is disabled
* 01/01/0001 for Date/Time ports if PMServer 4.0 date handling compatibility is enabled

Therefore, you can use variables as counters, which need an initial value. For example, you can create a numeric variable with the following expression:

VAR1 + 1

This expression counts the number of records in the VAR1 port. If the initial value of the variable were set to NULL, the expression would always evaluate to NULL. This is why the initial value is set to zero.

**Using Default Values for Ports:**

All transformations use default values that determine how the Informatica Server handles input null values and output transformation errors. Input, output, and input/output ports are created with a system default value that you can sometimes override with a user-defined default value. Default values have different functions in different types of ports.

* **Input port.** The system default value for null input ports is NULL. It displays as a blank in the transformation. If an input value is NULL, the Informatica Server leaves it as NULL.
* **Output port.** The system default value for output transformation errors is ERROR. The default value displays in the transformation as ERROR (‘transformation error’). If a transformation error occurs, the Informatica Server skips the row. The Informatica Server notes all input rows skipped by the ERROR function in the session log file. The following errors are considered transformation errors:
  + Data conversion errors, such as passing a number to a date function.
  + Expression evaluation errors, such as dividing by zero.
  + Calls to an ERROR function.
* **Input/output port.** The system default value for null input is the same as input ports, NULL. It displays as blank in the transformation. The default value for output transformation errors is the same as output ports. It does not display in the transformation.

**Entering User-Defined Default Values:**

You can override the system default values with user-defined default values for supported input, input/output, and output ports within a connected transformation.

* **Input ports.** You can enter user-defined default values for input ports if you do not want the Informatica Server to treat null values as NULL.
* **Output ports.** You can enter user-defined default values for output ports if you do not want the Informatica Server to skip the row, or if you want the Informatica Server to write a specific message with the skipped row to the session log.
* **Input/output ports.** You can enter user-defined default values to handle null input values for input/output ports in the same way you can enter user-defined default values for null input values for input ports. You cannot enter user-defined default values for output transformation errors in an input/output port.

**Entering Constant Values:**

You can enter any constant value as a default value. The constant value must match the port data type. For example, a default value for a numeric port must be a numeric constant. Some constant values include:

0

9999

NULL

‘Unknown Value’

‘Null input data’

**Entering Constant Expressions:**

A constant expression is any expression that uses transformation functions (except aggregate functions) to write constant expressions. You cannot use values from input, input/output, or variable ports in a constant expression. Some valid constant expressions include:

500 \* 1.75

TO\_DATE (‘January 1, 1998, 12:05 AM’)

ERROR (‘Null not allowed’)

ABORT (‘Null not allowed’)

SYSDATE

You cannot use values from ports within the expression because the Informatica Server assigns default values for the entire mapping when it initializes the session. Some invalid default values include the following examples, which incorporate values read from ports:

AVG (IN\_SALARY)

IN\_PRICE \* IN\_QUANTITY

: LKP (LKP\_DATES, DATE\_SHIPPED)

**Entering ERROR and ABORT Functions:**

You can use the ERROR and ABORT functions for input and output port default values, as well as input values for input/output ports. The Informatica Server skips the row when it encounters the ERROR function. It aborts the session when it encounters the ABORT function.

**Entering User-Defined Default Input Values:**

You can enter a user-defined default input value if you do not want the Informatica Server to treat null values as NULL. You can perform the following functions to override null values:

* Replace the null value with a constant value or constant expression.
* Skip the null value with an ERROR function.
* Abort the session with the ABORT function.

**Skipping Null Records:**

Use the ERROR function as the default value when you do not want null values to pass into a transformation. For example, you might want to skip a row when the input value of DEPT\_NAME is NULL. You could use the following expression as the default value:

ERROR(‘Error. DEPT is NULL’)

When you use the ERROR function as a default value, the Informatica Server skips the row with the null value. The Informatica Server writes all rows skipped by the ERROR function into the session log file. It does not write these rows to the session reject file.

**Entering User-Defined Default Output Values:**

You can enter user-defined default values for output ports if you do not want the Informatica Server to skip rows with errors or if you want the Informatica Server to write a specific message with the skipped row to the session log. You can enter default values to perform the following functions when the Informatica Server encounters output transformation errors:

* Replace the error with a constant value or constant expression. The Informatica Server does not skip the row.
* Abort the session with the ABORT function.
* Write specific messages in the session log for transformation errors.

**Configuring Tracing Levels in Transformations:**

When you configure a transformation, you can set the amount of detail the Informatica Server writes in the session log.

|  |  |
| --- | --- |
| Session Log Tracing Levels: | |
| **Tracing Level** | **Description** |
| Normal | Informatica Server logs initialization and status information; errors encountered, and skipped rows due to transformation row errors. Summarizes session results, but not at the level of individual rows. |
| Terse | Informatica Server logs initialization information as well as error messages and notification of rejected data. |
| Verbose Init | In addition to normal tracing, Informatica Server logs additional initialization details; names of index and data files used, and detailed transformation statistics. |
| Verbose Data | In addition to verbose initialization tracing, Informatica Server logs each row that passes into the mapping. Also notes where the Informatica Server truncates string data to fit the precision of a column and provides detailed transformation statistics. |

By default, the tracing level for every transformation is Normal. You change the tracing level to a Verbose setting only when you need to debug a transformation that is not behaving as expected. To add a slight performance boost, you can also set the tracing level to Terse, writing the minimum of detail to the session log when running a session containing the transformation. When you configure a session, you can override the tracing levels for individual transformations with a single tracing level for all transformations in the session.

**Connecting Ports:**

After you add transformations to a mapping or mapplet, you can connect the ports between transformations. The Informatica Server passes data from one transformation to another through the ports. You can connect many transformations to one transformation, or you can connect many to one, or one to one. When you connect transformations, the Designer performs validation. If the attempted connection is invalid, the Designer does not create the connection, and it displays an error message in the status bar explaining the problem.

**Copying and Linking Ports**

You can manually connect ports, or you can automatically link ports between two transformations or mapping objects.

**Manually Link**

To connect ports manually, choose Layout-Link Columns.

* When you drag from one port to another, the Designer creates a connection.
* If you drag into an empty port, the Designer copies the port and creates a connection.

You can also connect multiple ports at the same time. When you hold down the Ctrl or the Shift key while clicking ports, you can select a range to connect to another transformation. The Designer connects the ports, beginning with the top pair. It connects all ports that meet the connection validation requirements.

**Automatically Link**

To automatically link all ports between transformations, choose Layout-Autolink and select the mapping objects. For mapping objects that contain groups such as Router transformations or XML targets, select the group name. You can select to link by name or by position.

* If you select by name, the Designer adds links between input and output ports that have the same name.
* If you select by position, the Designer links the first output port to the first input port, the second output port to the second input port, and so forth.

**Connection Options:**

When you connect transformations, you can connect with the following options:

* **One to one.** Connect one transformation to one transformation or target only.
* **One to many.**
  + Connect one port to multiple transformations or targets.
  + Connect multiple ports in one transformation to multiple transformations or targets.
* **Many to one.** Connect many transformations to one transformation or target.

**Connecting One to Many:**

When you want to use the same data for different purposes, you can connect the port providing that data to several ports in the mapping. For example, salary information might be used to calculate the average salary in a department through the Aggregator transformation, while the same information might appear in an Expression transformation configured to calculate each employee’s monthly pay.

**Connecting Many to One:**

Frequently, you need to combine data from multiple transformations into a single transformation or target. For example, you might need to combine data from multiple Aggregator and Expression transformations in a single fact table in a star schema.

**Connection Validation:**

If the Designer detects an error when you try to connect ports, it displays a symbol indicating that you cannot connect the ports. You cannot connect ports that do not meet the following validation requirements:

* **Follow logic of data flow in the mapping.** Given the logic of the data flow between sources and targets, you can connect the following types of ports:
  + The receiving port must be an input or input/output port.
  + The originating port must be an output or input/output port.
  + You cannot connect ports in the same transformation.
* **The receiving transformation can have only one originating active transformation.** You cannot connect two active transformations to the same transformation, since the Informatica Server cannot verify that both transformations provide the same number of records. You can connect any number of passive transformations, or connect one active transformation with any number of passive transformations, to the same transformation or target.
* **The data types must be compatible.** The Designer checks if it can map between the two data types before connecting them. The Informatica Server cannot transform data between ports with incompatible data types. While the data types do not have to be identical, they do have to be compatible, such as Char and Varchar.
* **Connect a source definition to a Source Qualifier transformation only.** Every source must connect to a Source Qualifier or Normalizer transformation. You then connect the Source Qualifier or Normalizer to targets or other transformations.
* **Link columns to a target definition.** You can connect ports into a target definition in a mapping, but you cannot copy columns into a target definition in a mapping. Use the Warehouse Designer to add columns to a target definition.

**Reusable Transformations:**

Mappings can contain standard and reusable transformations. Standard transformations exist within a single mapping. Reusable transformations can be used in multiple mappings. For example, you might create an Expression transformation that calculates value-added tax for sales in Canada, which is useful when you analyze the cost of doing business in that country. Rather than perform the same work every time, you can create a reusable transformation. When you need to incorporate this transformation into a mapping, you add an instance of it to the mapping. Later, if you change the definition of the transformation, all instances of it inherit the changes.

The Designer stores each reusable transformation as metadata separate from any mappings that use the transformation. If you review the contents of a folder in the Navigator, you see the list of all reusable transformations in that folder. Each reusable transformation falls within a category of transformations available in the Designer. For example, you can create a reusable Aggregator transformation to perform the same aggregate calculations in multiple mappings, or a reusable Stored Procedure transformation to call the same stored procedure in multiple mappings. Normally, you have the option to create any transformation as a standard or reusable transformation. However, you can only create External Procedure and Advanced External Procedure transformations as reusable transformations. When you add instances of a reusable transformation to mappings, you must be careful about the changes you make to the transformation.

**Instances and Inherited Changes:**

When you add a reusable transformation to a mapping, you add an instance of the transformation. The definition of the transformation still exists outside the mapping, while a copy (or instance) appears within the mapping. Since the instance of a reusable transformation is a pointer to that transformation, when you change the transformation in the Transformation Developer, its instances automatically reflect these changes. This feature can save you a great deal of work. Instead of updating the same transformation in every mapping that uses it, you can update the reusable transformation once, and all instances of the transformation automatically inherit the change.

**Creating Reusable Transformations:**

You can create a reusable transformation using the following methods:

* **Design it in the Transformation Developer.** In the Transformation Developer, you can build new reusable transformations.
* **Promote a standard transformation from the Mapping Designer.** After you add a transformation to a mapping, you can promote it to the status of reusable transformation. The transformation designed in the mapping then becomes an instance of a reusable transformation maintained elsewhere in the repository.

Once you promote a transformation to reusable status, you can demote it to a standard transformation at any time. **Note:** Sequence Generator transformations must be reusable in mapplets. You cannot demote reusable Sequence Generator transformations to standard in a mapplet.As in the Mapping Designer; the Transformations toolbar appears in the Transformation Developer. To create a reusable transformation, however, you click and drag to create independent transformations that are not part of a mapping.

To create a reusable transformation:

* In the Designer, switch to the Transformation Developer.
* Click the button on the Transformation toolbar corresponding to the type of transformation you want to create.
* Click and drag within the workbook to create the transformation.
* Double-click the transformation title bar to open the dialog displaying its properties.
* Click the Rename button and enter a descriptive name for the transformation, and click OK. The naming convention for reusable transformations is reu*Transformation\_Name*.
* Click the Ports tab, then add any input and output ports you need for this transformation.
* Set the other properties of the transformation. These properties vary according to the transformation you create. For example, if you create an Expression transformation, you need to enter an expression for one or more of the transformation output ports. If you create a Stored Procedure transformation, you need to identify the stored procedure to call.
* Click OK.
* Choose Repository-Save.

**Promoting Standard Transformations:**

The other technique for creating a reusable transformation is to promote an existing transformation within a mapping. By checking the Make reusable option in the Edit Transformations dialog box, you instruct the Designer to promote the transformation and create an instance of it in the mapping.

To promote a standard transformation:

* In the Designer, open a mapping and double-click the title bar of the transformation you want to promote.
* Check the Make Reusable option.
* When prompted whether you are sure you want to promote the transformation, click Yes.
* Click OK to return to the mapping.
* Choose Repository-Save.

Now, when you look at the list of reusable transformations in the folder you are working in, the newly promoted transformation appears in this list.

**Creating Standard Instances of Reusable Transformations:**

You can create a standard, or non-reusable, instance of a reusable transformation within a mapping. Reusable transformations must be made standard within the same folder. If you want to have a standard instance of a reusable transformation in a different folder, you need to first make a standard instance of the transformation in the source folder, then copy it into the target folder. To create a standard instance of a reusable transformation:

* In the Designer, open a mapping and drag an existing transformation from the Navigator into the mapping. The status bar displays the following message: “Make a non-reusable copy of this transformation and add it to this mapping.”
* Drop the transformation into the mapping. You have now created a non-reusable or standard transformation from an existing reusable transformation.
* Choose Repository-Save.

**Adding Reusable Transformations to Mappings:**

Once you have created a reusable transformation, you can add it to mappings.

To add a reusable transformation:

* In the Designer, switch to the Mapping Designer.
* Open or create a mapping.
* In the list of repository objects, drill down until you find the reusable transformation you want in the Transformations section of a folder.
* Click and drag the transformation from the Navigator into the mapping. A copy (or instance) of the reusable transformation appears.
* Connect the new transformation to other transformations or target definitions as needed.
* Choose Repository-Save.

**Modifying a Reusable Transformation:**

Changes to a reusable transformation that you enter through the Transformation Developer are immediately reflected in all instances of that transformation. While this feature is a powerful way to save work and enforce standards (for example, by publishing the official version of a depreciation calculation through a reusable transformation), you risk invalidating mappings when you modify a reusable transformation. If you make any of the following changes to the reusable transformation, mappings that use instances of it may be invalidated:

* When you delete a port or multiple ports in a transformation, you disconnect the instance from part or all of the data flow through the mapping.
* When you change a port data type, you make it impossible to map data from that port to another port using an incompatible data type.
* When you change a port name, expressions that refer to the port are no longer valid.
* When you enter an invalid expression in the reusable transformation, mappings that use the transformation are no longer valid. The Informatica Server cannot run sessions based on invalid mappings until you correct the errors.

# Data Warehousing - ETL Project Life Cycle (Simple to understand)

-> Data warehousing projects are categorized into 4 types.

1) Development Projects.  
2) Enhancement Projects  
3) Migration Projects  
4) Production support Projects.

-> The following are the different phases involved in a ETL project development life cycle.

1) Business Requirement Collection ( BRD )  
2) System Requirement Collection ( SRD )  
3) Design Phase  
a) High Level Design Document ( HRD )  
b) Low level Design Document ( LLD )  
c) Mapping Design   
4) Code Review  
5) Peer Review  
6) Testing  
a) Unit Testing  
b) System Integration Testing.  
c) User Acceptance Testing (UAT)   
7) Pre - Production  
8) Production (Go-Live)

Business Requirement Collection: -   
---------------------------------------------

-> The business requirement gathering starts by business Analyst, onsite technical lead and client business users.

-> In this phase, a Business Analyst prepares Business Requirement Document (BRD) (or) Business Requirement Specifications (BRS)

-> BR collection takes place at client location.

-> The o/p from BR Analysis is

-> BRS: - Business Analyst will gather the Business Requirement and document in BRS

-> SRS: - senior technical people (or) ETL architect will prepare the SRS which contains s/w and h/w requirements.

The SRS will includes  
a) O/S to be used (windows or UNIX)   
b) RDBMS required to build database (oracle, Teradata etc.)  
c) ETL tools required (Informatica, Data stage)   
d) OLAP tools required ( Cognos ,BO )

The SRS is also called as Technical Requirement Specifications (TRS)

Designing and Planning the solutions: -   
------------------------------------------------

->the o/p from design and planning phase is  
a) HLD (High Level Design) Document  
b) LLD (Low Level Design) Document

HLD (High Level Design) Document: -

An ETL Architect and DWH Architect participate in designing a solution to build a DWH.  
An HLD document is prepared based on Business Requirement.

LLD (Low Level Design) Document: -  
  
Based on HLD,a senior ETL developer prepare Low Level Design Document   
The LLD contains more technical details of an ETL System.  
An LLD contains data flow diagram (DFD), details of source and targets of each mapping.  
An LLD also contains information about full and incremental load.  
After LLD then Development Phase will start

Development Phase (Coding):-  
--------------------------------------------------

-> Based on LLD, the ETL team will create mapping (ETL Code)   
-> after designing the mappings, the code (Mappings) will be reviewed by developers.

Code Review:-

-> Code Review will be done by developer.  
-> In code review, the developer will review the code and the logic but not the data.  
-> The following activities takes place in code review  
-> You have to check the naming standards of transformation, mappings of data etc.  
-> Source and target mapping ( Placed the correct logic or not in mapping )

Peer Review:-

-> The code will reviewed by your team member (third party developer)

Testing:-  
--------------------------------

The following various types testing carried out in testing environment.  
1) Unit Testing.  
2) Development Integration Testing.  
3) System Integration Testing.  
4) User Acceptance Testing.

Unit Testing:-

-> A unit test for the DWH is a white Box testing, It should check the ETL procedure and Mappings.  
-> The following are the test cases can be executed by an ETL developer.  
1) Verify data loss  
2) Number of records in the source and target  
3) Data load/Insert  
4) Data load/Update  
5) Incremental load  
6) Data accuracy  
7) verify Naming standards.  
8) Verify column Mapping

-> The Unit Test will be carried by ETL developer in development phase.  
-> ETL developer has to do the data validations also in this phase.

Development Integration Testing -

-> Run all the mappings in the sequence order.  
-> First run the source to stage mappings.  
-> Then run the mappings related to dimensions and facts.

System Integration Testing:-

-> After development phase, we have to move our code to QA environment.  
-> In this environment, we are giving read-only permission to testing people.  
-> They will test all the workflows.  
-> And they will test our code according to their standards.

User Acceptance Testing (UAT):-

-> This test is carried out in the presence of client side technical users to verify the data migration from source to destination.

Production Environment:-  
---------------------------------

->Migrate the code into the Go-Live environment from test environment (QA Environment).