**Lookup Transformation**

**1.** What is a lookup transformation?  
A lookup transformation is used to look up data in a flat file, relational table, view, and synonym.

**2.** What are the tasks of a lookup transformation?  
The lookup transformation is used to perform the following tasks?

* Get a related value: Retrieve a value from the lookup table based on a value in the source.
* Perform a calculation: Retrieve a value from a lookup table and use it in a calculation.
* Update slowly changing dimension tables: Determine whether rows exist in a target.

**3.** How do you configure a lookup transformation?  
Configure the lookup transformation to perform the following types of lookups:

* Relational or flat file lookup
* Pipeline lookup
* Connected or unconnected lookup
* Cached or uncached lookup

**4.** What is a pipeline lookup transformation?  
A pipeline lookup transformation is used to perform lookup on application sources such as JMS, MSMQ or SAP. A pipeline lookup transformation has a source qualifier as the lookups source.

**5.** What is connected and unconnected lookup transformation?

* A connected lookup transformation is connected the transformations in the mapping pipeline. It receives source data, performs a lookup and returns data to the pipeline.
* An unconnected lookup transformation is not connected to the other transformations in the mapping pipeline. A transformation in the pipeline calls the unconnected lookup with a :LKP expression.

**6.** What are the differences between connected and unconnected lookup transformation?

* Connected lookup transformation receives input values directly from the pipeline. Unconnected lookup transformation receives input values from the result of a :LKP expression in another transformation.
* Connected lookup transformation can be configured as dynamic or static cache. Unconnected lookup transformation can be configured only as static cache.
* Connected lookup transformation can return multiple columns from the same row or insert into the dynamic lookup cache. Unconnected lookup transformation can return one column from each row.
* If there is no match for the lookup condition, connected lookup transformation returns default value for all output ports. If you configure dynamic caching, the Integration Service inserts rows into the cache or leaves it unchanged. If there is no match for the lookup condition, the unconnected lookup transformation returns null.
* In a connected lookup transformation, the cache includes the lookup source columns in the lookup condition and the lookup source columns that are output ports. In an unconnected lookup transformation, the cache includes all lookup/output ports in the lookup condition and the lookup/return port.
* Connected lookup transformation passes multiple output values to another transformation. Unconnected lookup transformation passes one output value to another transformation.
* Connected lookup transformation supports user-defined values. Unconnected lookup transformation does not support user-defined default values.

**7.** How do you handle multiple matches in lookup transformation? or what is "Lookup Policy on Multiple Match"?  
"Lookup Policy on Multiple Match" option is used to determine which rows that the lookup transformation returns when it finds multiple rows that match the lookup condition. You can select lookup to return first or last row or any matching row or to report an error.

**8.** What is "Output Old Value on Update"?  
This option is used when dynamic cache is enabled. When this option is enabled, the integration service outputs old values out of the lookup/output ports. When the Integration Service updates a row in the cache, it outputs the value that existed in the lookup cache before it updated the row based on the input data. When the Integration Service inserts a new row in the cache, it outputs null values. When you disable this property, the Integration Service outputs the same values out of the lookup/output and input/output ports.

**9.** What is "Insert Else Update" and "Update Else Insert"?  
These options are used when dynamic cache is enabled.

* Insert Else Update option applies to rows entering the lookup transformation with the row type of insert. When this option is enabled the integration service inserts new rows in the cache and updates existing rows when disabled, the Integration Service does not update existing rows.
* Update Else Insert option applies to rows entering the lookup transformation with the row type of update. When this option is enabled, the Integration Service updates existing rows, and inserts a new row if it is new. When disabled, the Integration Service does not insert new rows.

**10.** What are the options available to configure a lookup cache?  
The following options can be used to configure a lookup cache:

* Persistent cache
* Recache from lookup source
* Static cache
* Dynamic cache
* Shared Cache
* Pre-build lookup cache

**11.** What is a cached lookup transformation and uncached lookup transformation?

* Cached lookup transformation: The Integration Service builds a cache in memory when it processes the first row of data in a cached Lookup transformation. The Integration Service stores condition values in the index cache and output values in the data cache. The Integration Service queries the cache for each row that enters the transformation.
* Uncached lookup transformation: For each row that enters the lookup transformation, the Integration Service queries the lookup source and returns a value. The integration service does not build a cache.

**12.** How the integration service builds the caches for connected lookup transformation?  
The Integration Service builds the lookup caches for connected lookup transformation in the following ways:

* Sequential cache: The Integration Service builds lookup caches sequentially. The Integration Service builds the cache in memory when it processes the first row of the data in a cached lookup transformation.
* Concurrent caches: The Integration Service builds lookup caches concurrently. It does not need to wait for data to reach the Lookup transformation.

**13.** How the integration service builds the caches for unconnected lookup transformation?  
The Integration Service builds caches for unconnected Lookup transformations as sequentially.

**14.** What is a dynamic cache?  
The dynamic cache represents the data in the target. The Integration Service builds the cache when it processes the first lookup request. It queries the cache based on the lookup condition for each row that passes into the transformation. The Integration Service updates the lookup cache as it passes rows to the target. The integration service either inserts the row in the cache or updates the row in the cache or makes no change to the cache.

**15.** When you use a dynamic cache, do you need to associate each lookup port with the input port?  
Yes. You need to associate each lookup/output port with the input/output port or a sequence ID. The Integration Service uses the data in the associated port to insert or update rows in the lookup cache.

**16.** What are the different values returned by NewLookupRow port?  
The different values are

* 0 - Integration Service does not update or insert the row in the cache.
* 1 - Integration Service inserts the row into the cache.
* 2 - Integration Service updates the row in the cache.

**17.** What is a persistent cache?  
If the lookup source does not change between session runs, then you can improve the performance by creating a persistent cache for the source. When a session runs for the first time, the integration service creates the cache files and saves them to disk instead of deleting them. The next time when the session runs, the integration service builds the memory from the cache file.

**18.** What is a shared cache?  
You can configure multiple Lookup transformations in a mapping to share a single lookup cache. The Integration Service builds the cache when it processes the first Lookup transformation. It uses the same cache to perform lookups for subsequent Lookup transformations that share the cache.

**19.** What is unnamed cache and named cache?

* Unnamed cache: When Lookup transformations in a mapping have compatible caching structures, the Integration Service shares the cache by default. You can only share static unnamed caches.
* Named cache: Use a persistent named cache when you want to share a cache file across mappings or share a dynamic and a static cache. The caching structures must match or be compatible with a named cache. You can share static and dynamic named caches.

**20.** How do you improve the performance of lookup transformation?

* Create an index on the columns used in the lookup condition
* Place conditions with equality operator first
* Cache small lookup tables.
* Join tables in the database: If the source and the lookup table are in the same database, join the tables in the database rather than using a lookup transformation.
* Use persistent cache for static lookups.
* Avoid ORDER BY on all columns in the lookup source. Specify explicitly the ORDER By clause on the required columns.
* For flat file lookups, provide Sorted files as lookup source.

**Aggregator Transformation**

**1.** What is aggregator transformation?  
Aggregator transformation performs aggregate calculations like sum, average, count etc. It is an active transformation, changes the number of rows in the pipeline. Unlike expression transformation (performs calculations on a row-by-row basis), an aggregator transformation performs calculations on group of rows.  
  
**2.** What is aggregate cache?  
The integration service creates index and data cache in memory to process the aggregator transformation and stores the data group in index cache, row data in data cache. If the integration service requires more space, it stores the overflow values in cache files.  
  
**3.** How can we improve performance of aggregate transformation?

* Use sorted input: Sort the data before passing into aggregator. The integration service uses memory to process the aggregator transformation and it does not use cache memory.
* Filter the unwanted data before aggregating.
* Limit the number of input/output or output ports to reduce the amount of data the aggregator transformation stores in the data cache.

**4.** What are the different types of aggregate functions?  
  
The different types of aggregate functions are listed below:

* AVG
* COUNT
* FIRST
* LAST
* MAX
* MEDIAN
* MIN
* PERCENTILE
* STDDEV
* SUM
* VARIANCE

**5.** Why cannot you use both single level and nested aggregate functions in a single aggregate transformation?  
  
The nested aggregate function returns only one output row, whereas the single level aggregate function returns more than one row. Since the number of rows returned are not same, you cannot use both single level and nested aggregate functions in the same transformation. If you include both the single level and nested functions in the same aggregator, the designer marks the mapping or mapplet as invalid. So, you need to create separate aggregator transformations.  
  
**6.** Up to how many levels, you can nest the aggregate functions?  
  
We can nest up to two levels only.  
Example: MAX( SUM( ITEM ) )  
  
**7.** What is incremental aggregation?  
  
The integration service performs aggregate calculations and then stores the data in historical cache. Next time when you run the session, the integration service reads only new data and uses the historical cache to perform new aggregation calculations incrementally.  
  
**8.** Why cannot we use sorted input option for incremental aggregation?  
  
In incremental aggregation, the aggregate calculations are stored in historical cache on the server. In this historical cache the data need not be in sorted order.  If you give sorted input, the records come as presorted for that particular run but in the historical cache the data may not be in the sorted order. That is why this option is not allowed.  
  
**9.** How the NULL values are handled in Aggregator?  
  
You can configure the integration service to treat null values in aggregator functions as NULL or zero. By default the integration service treats null values as NULL in aggregate functions.

# Informatica Scenario Based Interview Questions with Answers - Part 1

### Informatica Scenarios

I have listed the following **informatica scenarios** which are frequently asked in the informatica interviews. These **informatica scenario interview questions** helps you a lot in gaining confidence in interviews.  
  
**1.**How to generate sequence numbers using expression transformation?  
  
**Solution:**  
In the expression transformation, create a variable port and increment it by 1. Then assign the variable port to an output port. In the expression transformation, the ports are:  
V\_count=V\_count+1  
O\_count=V\_count  
  
**2.** Design a mapping to load the first 3 rows from a flat file into a target?  
  
**Solution:**  
You have to assign row numbers to each record. Generate the row numbers either using the expression transformation as mentioned above or use sequence generator transformation.  
Then pass the output to filter transformation and specify the filter condition as O\_count <=3  
  
**3.** Design a mapping to load the last 3 rows from a flat file into a target?  
  
**Solution:**  
Consider the source has the following data.  
col  
a  
b  
c  
d  
e  
  
**Step1:** You have to assign row numbers to each record. Generate the row numbers using the expression transformation as mentioned above and call the row number generated port as O\_count. Create a DUMMY output port in the same expression transformation and assign 1 to that port. So that, the DUMMY output port always return 1 for each row.  
  
In the expression transformation, the ports are  
V\_count=V\_count+1  
O\_count=V\_count  
O\_dummy=1  
  
The output of expression transformation will be  
col, o\_count, o\_dummy  
a, 1, 1  
b, 2, 1  
c, 3, 1  
d, 4, 1  
e, 5, 1  
  
**Step2:** Pass the output of expression transformation to aggregator and do not specify any group by condition. Create an output port O\_total\_records in the aggregator and assign O\_count port to it. The aggregator will return the last row by default. The output of aggregator contains the DUMMY port which has value 1 and O\_total\_records port which has the value of total number of records in the source.  
  
In the aggregator transformation, the ports are  
O\_dummy  
O\_count  
O\_total\_records=O\_count  
  
The output of aggregator transformation will be  
O\_total\_records, O\_dummy  
5, 1  
  
**Step3:** Pass the output of expression transformation, aggregator transformation to joiner transformation and join on the DUMMY port. In the joiner transformation check the property sorted input, then only you can connect both expression and aggregator to joiner transformation.  
  
In the joiner transformation, the join condition will be  
O\_dummy (port from aggregator transformation) = O\_dummy (port from expression transformation)  
  
The output of joiner transformation will be  
col, o\_count, o\_total\_records  
a, 1, 5  
b, 2, 5  
c, 3, 5  
d, 4, 5  
e, 5, 5  
  
**Step4:** Now pass the ouput of joiner transformation to filter transformation and specify the filter condition as O\_total\_records (port from aggregator)-O\_count(port from expression) <=2  
  
In the filter transformation, the filter condition will be  
O\_total\_records - O\_count <=2  
  
The output of filter transformation will be  
col o\_count, o\_total\_records  
c, 3, 5  
d, 4, 5  
e, 5, 5  
  
**4.** Design a mapping to load the first record from a flat file into one table A, the last record from a flat file into table B and the remaining records into table C?   
  
**Solution:**  
This is similar to the above problem; the first 3 steps are same. In the last step instead of using the filter transformation, you have to use router transformation. In the router transformation create two output groups.  
  
In the first group, the condition should be O\_count=1 and connect the corresponding output group to table A. In the second group, the condition should be O\_count=O\_total\_records and connect the corresponding output group to table B. The output of default group should be connected to table C.  
  
**5.** Consider the following products data which contain duplicate rows.  
A  
B  
C  
C  
B  
D  
B  
  
**Q1.** Design a mapping to load all unique products in one table and the duplicate rows in another table.  
The first table should contain the following output  
A  
D  
  
The second target should contain the following output  
B  
B  
B  
C  
C  
  
**Solution:**  
Use sorter transformation and sort the products data. Pass the output to an expression transformation and create a dummy port O\_dummy and assign 1 to that port. So that, the DUMMY output port always return 1 for each row.  
  
The output of expression transformation will be  
Product, O\_dummy  
A, 1  
B, 1  
B, 1  
B, 1  
C, 1  
C, 1  
D, 1  
  
Pass the output of expression transformation to an aggregator transformation. Check the group by on product port. In the aggreagtor, create an output port O\_count\_of\_each\_product and write an expression count(product).  
  
The output of aggregator will be  
Product, O\_count\_of\_each\_product  
A, 1  
B, 3  
C, 2  
D, 1  
  
Now pass the output of expression transformation, aggregator transformation to joiner transformation and join on the products port. In the joiner transformation check the property sorted input, then only you can connect both expression and aggregator to joiner transformation.  
  
The output of joiner will be  
product, O\_dummy, O\_count\_of\_each\_product  
A, 1, 1  
B, 1, 3  
B, 1, 3  
B, 1, 3  
C, 1, 2  
C, 1, 2  
D, 1, 1  
  
Now pass the output of joiner to a router transformation, create one group and specify the group condition as O\_dummy=O\_count\_of\_each\_product. Then connect this group to one table. Connect the output of default group to another table.  
  
**Q2**. Design a mapping to load each product once into one table and the remaining products which are duplicated into another table.  
The first table should contain the following output  
A  
B  
C  
D  
  
The second table should contain the following output  
B  
B  
C  
  
**Solution:**  
Use sorter transformation and sort the products data. Pass the output to an expression transformation and create a variable port,V\_curr\_product, and assign product port to it. Then create a V\_count port and in the expression editor write IIF(V\_curr\_product=V\_prev\_product, V\_count+1,1). Create one more variable port V\_prev\_port and assign product port to it. Now create an output port O\_count port and assign V\_count port to it.  
  
In the expression transformation, the ports are  
Product  
V\_curr\_product=product  
V\_count=IIF(V\_curr\_product=V\_prev\_product,V\_count+1,1)  
V\_prev\_product=product  
O\_count=V\_count  
  
The output of expression transformation will be  
Product, O\_count  
A, 1  
B, 1  
B, 2  
B, 3  
C, 1  
C, 2  
D, 1  
  
Now Pass the output of expression transformation to a router transformation, create one group and specify the condition as O\_count=1. Then connect this group to one table. Connect the output of default group to another table.

# Informatica Scenario Based Questions - Part 2

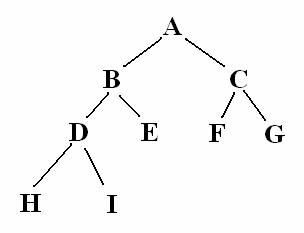
**1.** Consider the following employees data as source  
  
employee\_id, salary  
10, 1000  
20, 2000  
30, 3000  
40, 5000  
  
  
**Q1.** Design a mapping to load the cumulative sum of salaries of employees into target table?  
The target table data should look like as  
  
employee\_id, salary, cumulative\_sum  
10, 1000, 1000  
20, 2000, 3000  
30, 3000, 6000  
40, 5000, 11000  
  
**Solution:**  
  
Connect the source Qualifier to expression transformation. In the expression transformation, create a variable port V\_cum\_sal and in the expression editor write V\_cum\_sal+salary. Create an output port O\_cum\_sal and assign V\_cum\_sal to it.  
  
  
**Q2.** Design a mapping to get the pervious row salary for the current row. If there is no pervious row exists for the current row, then the pervious row salary should be displayed as null.  
The output should look like as  
  
employee\_id, salary, pre\_row\_salary  
10, 1000, Null  
20, 2000, 1000  
30, 3000, 2000  
40, 5000, 3000  
  
**Solution:**  
  
Connect the source Qualifier to expression transformation. In the expression transformation, create a variable port V\_count and increment it by one for each row entering the expression transformation. Also create V\_salary variable port and assign the expression IIF(V\_count=1,NULL,V\_prev\_salary) to it . Then create one more variable port V\_prev\_salary and assign Salary to it. Now create output port O\_prev\_salary and assign V\_salary to it. Connect the expression transformation to the target ports.  
  
In the expression transformation, the ports will be  
  
employee\_id  
salary  
V\_count=V\_count+1  
V\_salary=IIF(V\_count=1,NULL,V\_prev\_salary)  
V\_prev\_salary=salary  
O\_prev\_salary=V\_salary  
  
  
**Q3.** Design a mapping to get the next row salary for the current row. If there is no next row for the current row, then the next row salary should be displayed as null.  
The output should look like as  
  
employee\_id, salary, next\_row\_salary  
10, 1000, 2000  
20, 2000, 3000  
30, 3000, 5000  
40, 5000, Null  
  
**Solution:**  
  
**Step1:** Connect the source qualifier to two expression transformation. In each expression transformation, create a variable port V\_count and in the expression editor write V\_count+1. Now create an output port O\_count in each expression transformation. In the first expression transformation, assign V\_count to O\_count. In the second expression transformation assign V\_count-1 to O\_count.  
  
In the first expression transformation, the ports will be  
  
employee\_id   
salary  
V\_count=V\_count+1  
O\_count=V\_count  
  
In the second expression transformation, the ports will be  
  
employee\_id   
salary  
V\_count=V\_count+1  
O\_count=V\_count-1  
  
**Step2:** Connect both the expression transformations to joiner transformation and join them on the port O\_count. Consider the first expression transformation as Master and second one as detail. In the joiner specify the join type as Detail Outer Join. In the joiner transformation check the property sorted input, then only you can connect both expression transformations to joiner transformation.  
  
**Step3:** Pass the output of joiner transformation to a target table. From the joiner, connect the employee\_id, salary which are obtained from the first expression transformation to the employee\_id, salary ports in target table. Then from the joiner, connect the salary which is obtained from the second expression transformaiton to the next\_row\_salary port in the target table.  
  
  
**Q4.** Design a mapping to find the sum of salaries of all employees and this sum should repeat for all the rows.  
The output should look like as  
  
employee\_id, salary, salary\_sum  
10, 1000, 11000  
20, 2000, 11000  
30, 3000, 11000  
40, 5000, 11000  
  
**Solution:**  
  
**Step1:** Connect the source qualifier to the expression transformation. In the expression transformation, create a dummy port and assign value 1 to it.  
  
In the expression transformation, the ports will be  
  
employee\_id  
salary  
O\_dummy=1  
  
**Step2:** Pass the output of expression transformation to aggregator. Create a new port O\_sum\_salary and in the expression editor write SUM(salary). Do not specify group by on any port.  
  
In the aggregator transformation, the ports will be  
  
salary  
O\_dummy  
O\_sum\_salary=SUM(salary)  
  
**Step3:** Pass the output of expression transformation, aggregator transformation to joiner transformation and join on the DUMMY port. In the joiner transformation check the property sorted input, then only you can connect both expression and aggregator to joiner transformation.  
  
**Step4:** Pass the output of joiner to the target table.  
  
  
**2.**Consider the following employees table as source  
  
department\_no, employee\_name  
20, R  
10, A  
10, D  
20, P  
10, B  
10, C  
20, Q  
20, S  
  
  
**Q1.** Design a mapping to load a target table with the following values from the above source?  
  
department\_no, employee\_list  
10, A  
10, A,B  
10, A,B,C  
10, A,B,C,D  
20, A,B,C,D,P  
20, A,B,C,D,P,Q  
20, A,B,C,D,P,Q,R  
20, A,B,C,D,P,Q,R,S  
  
**Solution:**  
  
**Step1:** Use a sorter transformation and sort the data using the sort key as department\_no and then pass the output to the expression transformation. In the expression transformation, the ports will be  
  
department\_no  
employee\_name  
V\_employee\_list = IIF(ISNULL(V\_employee\_list),employee\_name,V\_employee\_list||','||employee\_name)  
O\_employee\_list = V\_employee\_list  
  
**Step2:** Now connect the expression transformation to a target table.  
  
  
**Q2.** Design a mapping to load a target table with the following values from the above source?  
  
department\_no, employee\_list  
10, A  
10, A,B  
10, A,B,C  
10, A,B,C,D  
20, P  
20, P,Q  
20, P,Q,R  
20, P,Q,R,S  
  
**Solution:**  
  
**Step1:** Use a sorter transformation and sort the data using the sort key as department\_no and then pass the output to the expression transformation. In the expression transformation, the ports will be  
  
department\_no  
employee\_name  
V\_curr\_deptno=department\_no  
V\_employee\_list = IIF(V\_curr\_deptno! = V\_prev\_deptno,employee\_name,V\_employee\_list||','||employee\_name)  
V\_prev\_deptno=department\_no  
O\_employee\_list = V\_employee\_list  
  
**Step2:** Now connect the expression transformation to a target table.  
  
  
**Q3.** Design a mapping to load a target table with the following values from the above source?  
  
department\_no, employee\_names  
10, A,B,C,D  
20, P,Q,R,S  
  
**Solution:**   
  
The first step is same as the above problem. Pass the output of expression to an aggregator transformation and specify the group by as department\_no. Now connect the aggregator transformation to a target table.

# Informatica Scenario Based Questions - Part 3

**1.** Consider the following product types data as the source.  
  
  
Product\_id, product\_type  
10, video  
10, Audio  
20, Audio  
30, Audio  
40, Audio  
50, Audio  
10, Movie  
20, Movie  
30, Movie  
40, Movie  
50, Movie  
60, Movie  
  
Assume that there are only 3 product types are available in the source. The source contains 12 records and you dont know how many products are available in each product type.  
  
  
**Q1.** Design a mapping to select 9 products in such a way that 3 products should be selected from video, 3 products should be selected from Audio and the remaining 3 products should be selected from Movie.  
  
**Solution:**  
  
**Step1:** Use sorter transformation and sort the data using the key as product\_type.  
  
**Step2:** Connect the sorter transformation to an expression transformation. In the expression transformation, the ports will be  
  
product\_id  
product\_type  
V\_curr\_prod\_type=product\_type  
V\_count = IIF(V\_curr\_prod\_type = V\_prev\_prod\_type,V\_count+1,1)  
V\_prev\_prod\_type=product\_type  
O\_count=V\_count  
  
**Step3:** Now connect the expression transformaion to a filter transformation and specify the filter condition as O\_count<=3. Pass the output of filter to a target table.  
  
  
**Q2.** In the above problem Q1, if the number of products in a particular product type are less than 3, then you wont get the total 9 records in the target table. For example, see the videos type in the source data. Now design a mapping in such way that even if the number of products in a particular product type are less than 3, then you have to get those less number of records from another porduc types. For example: If the number of products in videos are 1, then the reamaining 2 records should come from audios or movies. So, the total number of records in the target table should always be 9.  
  
**Solution:**  
  
The first two steps are same as above.  
  
**Step3:** Connect the expression transformation to a sorter transformation and sort the data using the key as O\_count. The ports in soter transformation will be  
  
product\_id  
product\_type  
O\_count (sort key)  
  
**Step3:** Discard O\_count port and connect the sorter transformation to an expression transformation. The ports in expression transformation will be  
  
product\_id  
product\_type  
V\_count=V\_count+1  
O\_prod\_count=V\_count  
  
**Step4:** Connect the expression to a filter transformation and specify the filter condition as O\_prod\_count<=9. Connect the filter transformation to a target table.  
  
  
**2.** Design a mapping to convert column data into row data without using the normalizer transformation.  
The source data looks like  
  
col1, col2, col3  
a, b, c  
d, e, f  
  
The target table data should look like  
  
Col  
a  
b  
c  
d  
e  
f  
  
**Solution:**  
  
Create three expression transformations with one port each. Connect col1 from Source Qualifier to port in first expression transformation. Connect col2 from Source Qualifier to port in second expression transformation. Connect col3 from source qualifier to port in third expression transformation. Create a union transformation with three input groups and each input group should have one port. Now connect the expression transformations to the input groups and connect the union transformation to the target table.  
  
  
**3.** Design a mapping to convert row data into column data.  
The source data looks like  
  
id, value  
10, a  
10, b  
10, c  
20, d  
20, e  
20, f  
  
The target table data should look like  
  
id, col1, col2, col3  
10, a, b, c  
20, d, e, f  
  
**Solution:**  
  
**Step1:** Use sorter transformation and sort the data using id port as the key. Then connect the sorter transformation to the expression transformation.  
  
**Step2:** In the expression transformation, create the ports and assign the expressions as mentioned below.  
  
id  
value  
V\_curr\_id=id  
V\_count= IIF(v\_curr\_id=V\_prev\_id,V\_count+1,1)  
V\_prev\_id=id  
O\_col1= IIF(V\_count=1,value,NULL)  
O\_col2= IIF(V\_count=2,value,NULL)  
O\_col3= IIF(V\_count=3,value,NULL)  
  
**Step3:** Connect the expression transformation to aggregator transformation. In the aggregator transforamtion, create the ports and assign the expressions as mentioned below.  
  
id (specify group by on this port)  
O\_col1  
O\_col2  
O\_col3  
col1=MAX(O\_col1)  
col2=MAX(O\_col2)  
col3=MAX(O\_col3)  
  
**Stpe4:** Now connect the ports id, col1, col2, col3 from aggregator transformation to the target table.

# Informatica Scenario Based Questions - Part 4

Take a look at the following tree structure diagram. From the tree structure, you can easily derive the parent-child relationship between the elements. For example, B is parent of D and E.

[](http://2.bp.blogspot.com/-nYr4S2bK_ts/ThQ9gseuN3I/AAAAAAAAALs/fc-Tyn-A_Ig/s1600/tree.jpg)

The above tree structure data is represented in a table as shown below.  
  
c1, c2, c3, c4  
A, B, D, H  
A, B, D, I  
A, B, E, NULL  
A, C, F, NULL  
A, C, G, NULL  
  
Here in this table, column C1 is parent of column C2, column C2 is parent of column C3, column C3 is parent of column C4.  
  
**Q1.** Design a mapping to load the target table with the below data. Here you need to generate sequence numbers for each element and then you have to get the parent id. As the element "A" is at root, it does not have any parent and its parent\_id is NULL.  
  
id, element, parent\_id  
1, A, NULL  
2, B, 1  
3, C, 1  
4, D, 2  
5, E, 2  
6, F, 3  
7, G, 3  
8, H, 4  
9, I, 4  
  
I have provided the solution for this problem in Oracle Sql query. If you are interested you can[Click Here](http://www.folkstalk.com/2011/11/oracle-complex-queries-part-3.html) to see the solution.  
  
**Q2.** This is an extension to the problem Q1. Let say column C2 has null for all the rows, then C1 becomes the parent of C3 and c3 is parent of C4. Let say both columns c2 and c3 has null for all the rows. Then c1 becomes the parent of c4. Design a mapping to accommodate these type of null conditions.

# Informatica Scenario Based Questions - Part 5

**Q1.** The source data contains only column 'id'. It will have sequence numbers from 1 to 1000. The source data looks like as

Id  
1  
2  
3  
4  
5  
6  
7  
8  
....  
1000

Create a workflow to load only the Fibonacci numbers in the target table. The target table data should look like as

 Id  
1  
2  
3  
5  
8  
13  
.....

In Fibonacci series each subsequent number is the sum of previous two numbers. Here assume that the first two numbers of the fibonacci series are 1 and 2.   
  
**Solution:**  
  
**STEP1:** Drag the source to the mapping designer and then in the Source Qualifier Transformation properties, set the number of sorted ports to one. This will sort the source data in ascending order. So that we will get the numbers in sequence as 1, 2, 3, ....1000  
  
**STEP2:** Connect the Source Qualifier Transformation to the Expression Transformation. In the Expression Transformation, create three variable ports and one output port. Assign the expressions to the ports as shown below.  
  
Ports in Expression Transformation:  
id  
v\_sum = v\_prev\_val1 + v\_prev\_val2  
v\_prev\_val1 = IIF(id=1 or id=2,1, IIF(v\_sum = id, v\_prev\_val2, v\_prev\_val1) )  
v\_prev\_val2 = IIF(id=1 or id =2, 2, IIF(v\_sum=id, v\_sum, v\_prev\_val2) )  
o\_flag = IIF(id=1 or id=2,1, IIF( v\_sum=id,1,0) )  
  
**STEP3:** Now connect the Expression Transformation to the Filter Transformation and specify the Filter Condition as o\_flag=1   
  
**STEP4:** Connect the Filter Transformation to the Target Table.  
  
  
**Q2.** The source table contains two columns "id" and "val". The source data looks like as below

id     val  
1      a,b,c  
2      pq,m,n  
3      asz,ro,liqt

Here the "val" column contains comma delimited data and has three fields in that column.  
Create a workflow to split the fields in “val” column to separate rows. The output should look like as below.

id     val  
1      a  
1      b  
1      c  
2      pq  
2      m  
2      n  
3      asz  
3      ro  
3      liqt

**Solution:**  
  
**STEP1:** Connect three Source Qualifier transformations to the Source Definition  
  
**STEP2:** Now connect all the three Source Qualifier transformations to the Union Transformation. Then connect the Union Transformation to the Sorter Transformation. In the sorter transformation sort the data based on Id port in ascending order.  
  
**STEP3:** Pass the output of Sorter Transformation to the Expression Transformation. The ports in Expression Transformation are:  
  
id (input/output port)  
val (input port)  
v\_currend\_id (variable port) = id  
v\_count (variable port) = IIF(v\_current\_id!=v\_previous\_id,1,v\_count+1)  
v\_previous\_id (variable port) = id  
o\_val (output port) = DECODE(v\_count, 1,  
        SUBSTR(val, 1, INSTR(val,',',1,1)-1 ),  
        2,  
        SUBSTR(val, INSTR(val,',',1,1)+1, INSTR(val,',',1,2)-INSTR(val,',',1,1)-1),  
        3,  
        SUBSTR(val, INSTR(val,',',1,2)+1),  
        NULL  
        )  
  
**STEP4:** Now pass the output of Expression Transformation to the Target definition. Connect id, o\_val ports of Expression Transformation to the id, val ports of Target Definition.

# Filter Transformation

**1.** What is a filter transformation?  
  
A filter transformation is used to filter out the rows in mapping. The filter transformation allows the rows that meet the filter condition to pass through and drops the rows that do not meet the condition. Filter transformation is an active transformation.  
  
**2.** Can we specify more than one filter condition in a filter transformation?  
  
We can only specify one condition in the filter transformation. To specify more than one condition, we have to use router transformation?  
  
**3.** In which case a filter transformation acts as passive transformation?  
  
If the filter condition is set to TRUE, then it passes all the rows without filtering any data. In this case, the filter transformation acts as passive transformation.  
  
**4.** Can we concatenate ports from more than one transformation into the filter transformation?  
  
No. The input ports for the filter must come from a single transformation.  
  
**5.** How to filter the null values and spaces?  
  
Use the ISNULL and IS\_SPACES functions  
Example: IIF(ISNULL(commission),FALSE,TRUE)  
  
**6.** How session performance can be improved by using filter transformation?  
  
Keep the filter transformation as close as possible to the sources in the mapping. This allows the unwanted data to be discarded and the integration service processes only the required rows. If the source is relational source, use the source qualifier to filter the rows.

**Joiner Transformation**

**1.** What is a joiner transformation?  
  
A joiner transformation joins two heterogeneous sources. You can also join the data from the same source. The joiner transformation joins sources with at least one matching column. The joiner uses a condition that matches one or more joins of columns between the two sources.  
  
**2.** How many joiner transformations are required to join n sources?  
  
To join n sources n-1 joiner transformations are required.  
  
**3.** What are the limitations of joiner transformation?

* You cannot use a joiner transformation when input pipeline contains an update strategy transformation.
* You cannot use a joiner if you connect a sequence generator transformation directly before the joiner.

**4.** What are the different types of joins?

* Normal join: In a normal join, the integration service discards all the rows from the master and detail source that do not match the join condition.
* Master outer join: A master outer join keeps all the rows of data from the detail source and the matching rows from the master source. It discards the unmatched rows from the master source.
* Detail outer join: A detail outer join keeps all the rows of data from the master source and the matching rows from the detail source. It discards the unmatched rows from the detail source.
* Full outer join: A full outer join keeps all rows of data from both the master and detail rows.

**5.** What is joiner cache?  
  
When the integration service processes a joiner transformation, it reads the rows from master source and builds the index and data cached. Then the integration service reads the detail source and performs the join. In case of sorted joiner, the integration service reads both sources (master and detail) concurrently and builds the cache based on the master rows.  
  
**6.** How to improve the performance of joiner transformation?

* Join sorted data whenever possible.
* For an unsorted Joiner transformation, designate the source with fewer rows as the master source.
* For a sorted Joiner transformation, designate the source with fewer duplicate key values as the master source.

**7.** Why joiner is a blocking transformation?  
  
When the integration service processes an unsorted joiner transformation, it reads all master rows before it reads the detail rows. To ensure it reads all master rows before the detail rows, the integration service blocks all the details source while it caches rows from the master source. As it blocks the detail source, the unsorted joiner is called a blocking transformation.  
  
**8.** What are the settings used to configure the joiner transformation

* Master and detail source
* Type of join
* Join condition

**Normalizer Transformation**

**1.** What is normalizer transformation?  
  
The normalizer transformation receives a row that contains multiple-occurring columns and retruns a row for each instance of the multiple-occurring data. This means it converts column data in to row data. Normalizer is an active transformation.  
  
**2.** Which transformation is required to process the cobol sources?  
  
Since the cobol sources contain denormalzed data, normalizer transformation is used to normalize the cobol sources.  
  
**3.** What is generated key and generated column id in a normalizer transformation?

* The integration service increments the generated key sequence number each time it process a source row. When the source row contains a multiple-occurring column or a multiple-occurring group of columns, the normalizer transformation returns a row for each occurrence. Each row contains the same generated key value.
* The normalizer transformation has a generated column ID (GCID) port for each multiple-occurring column. The GCID is an index for the instance of the multiple-occurring data. For example, if a column occurs 3 times in a source record, the normalizer returns a value of 1,2 or 3 in the generated column ID.

**4.** What is VSAM?  
  
VSAM (Virtual Storage Access Method) is a file access method for an IBM mainframe operating system. VSAM organize records in indexed or sequential flat files.  
  
**5.** What is VSAM normalizer transformation?  
  
The VSAM normalizer transformation is the source qualifier transformation for a COBOL source definition. A COBOL source is flat file that can contain multiple-occurring data and multiple types of records in the same file.  
  
**6.** What is pipeline normalizer transformation?  
  
Pipeline normalizer transformation processes multiple-occurring data from relational tables or flat files.  
  
**7.** What is occurs clause and redefines clause in normalizer transformation?

* Occurs clause is specified when the source row has a multiple-occurring columns.
* A redefines clause is specified when the source has rows of multiple columns.

# Rank Transformation

**1.** What is rank transformation?  
  
A rank transformation is used to select top or bottom rank of data. This means, it selects the largest or smallest numeric value in a port or group. Rank transformation also selects the strings at the top or bottom of a session sort order. Rank transformation is an active transformation.  
  
**2.** What is rank cache?  
  
The integration service compares input rows in the data cache, if the input row out-ranks a cached row, the integration service replaces the cached row with the input row. If you configure the rank transformation to rank across multiple groups, the integration service ranks incrementally for each group it finds. The integration service stores group information in index cache and row data in data cache.  
  
**3.** What is RANKINDEX port?  
  
The designer creates RANKINDEX port for each rank transformation. The integration service uses the rank index port to store the ranking position for each row in a group.  
  
**4.** How do you specify the number of rows you want to rank in a rank transformation?  
  
In the rank transformation properties, there is an option 'Number of Ranks' for specifying the number of rows you wants to rank.  
  
**5.** How to select either top or bottom ranking for a column?  
  
In the rank transformation properties, there is an option 'Top/Bottom' for selecting the top or bottom ranking for a column.  
  
**6.** Can we specify ranking on more than one port?  
  
No. We can specify to rank the data based on only one port. In the ports tab, you have to check the R option for designating the port as a rank port and this option can be checked only on one port.

**Router Transformation**

**1.** What is a router transformation?  
  
A router is used to filter the rows in a mapping. Unlike filter transformation, you can specify one or more conditions in a router transformation. Router is an active transformation.  
  
**2.** How to improve the performance of a session using router transformation?  
  
Use router transformation in a mapping instead of creating multiple filter transformations to perform the same task. The router transformation is more efficient in this case. When you use a router transformation in a mapping, the integration service processes the incoming data only once. When you use multiple filter transformations, the integration service processes the incoming data for each transformation.  
  
**3.** What are the different groups in router transformation?  
  
The router transformation has the following types of groups:

* Input
* Output

**4.** How many types of output groups are there?  
  
There are two types of output groups:

* User-defined group
* Default group

**5.** Where you specify the filter conditions in the router transformation?  
  
You can creat the group filter conditions in the groups tab using the expression editor.  
  
**6.** Can you connect ports of two output groups from router transformation to a single target?  
  
No. You cannot connect more than one output group to one target or a single input group transformation.

**Sequence Generator Transformation**

**1.** What is a sequence generator transformation?  
  
A Sequence generator transformation generates numeric values. Sequence generator transformation is a passive transformation.  
  
**2.** What is the use of a sequence generator transformation?  
  
A sequence generator is used to create unique primary key values, replace missing primary key values or cycle through a sequential range of numbers.  
  
**3.** What are the ports in sequence generator transformation?  
  
A sequence generator contains two output ports. They are CURRVAL and NEXTVAL.  
  
**4.** What is the maximum number of sequence that a sequence generator can generate?  
  
The maximum value is 9,223,372,036,854,775,807  
  
**5.** When you connect both the NEXTVAL and CURRVAL ports to a target, what will be the output values of these ports?  
  
The output values are  
NEXTVAL  CURRVAL  
1        2  
2        3  
3        4  
4        5  
5        6  
  
**6.** What will be the output value, if you connect only CURRVAL to the target without connecting NEXTVAL?  
  
The integration service passes a constant value for each row.  
  
**7.** What will be the value of CURRVAL in a sequence generator transformation?  
  
CURRVAL is the sum of "NEXTVAL" and "Increment By" Value.  
  
**8.** What is the number of cached values set to default for a sequence generator transformation?  
  
For non-reusable sequence generators, the number of cached values is set to zero.  
For reusable sequence generators, the number of cached values is set to 1000.  
  
**9.** How do you configure a sequence generator transformation?  
  
The following properties need to be configured for a sequence generator transformation:

* Start Value
* Increment By
* End Value
* Current Value
* Cycle
* Number of Cached Values

**Stored Procedure Transformation**

**1.** What is a stored procedure?  
  
A stored procedure is a precompiled collection of database procedural statements. Stored procedures are stored and run within the database.  
  
**2.** Give some examples where a stored procedure is used?  
  
The stored procedure can be used to do the following tasks

* Check the status of a target database before loading data into it.
* Determine if enough space exists in a database.
* Perform a specialized calculation.
* Drop and recreate indexes.

**3.** What is a connected stored procedure transformation?  
  
The stored procedure transformation is connected to the other transformations in the mapping pipeline.  
  
**4.** In which scenarios a connected stored procedure transformation is used?

* Run a stored procedure every time a row passes through the mapping.
* Pass parameters to the stored procedure and receive multiple output parameters.

**5.** What is an unconnected stored procedure transformation?  
  
The stored procedure transformation is not connected directly to the flow of the mapping. It either runs before or after the session or is called by an expression in another transformation in the mapping.  
  
**6.** In which scenarios an unconnected stored procedure transformation is used?

* Run a stored procedure before or after a session
* Run a stored procedure once during a mapping, such as pre or post-session.
* Run a stored procedure based on data that passes through the mapping, such as when a specific port does not contain a null value.
* Run nested stored procedures.
* Call multiple times within a mapping.

**7.** What are the options available to specify when the stored procedure transformation needs to be run?  
  
The following options describe when the stored procedure transformation runs:

* Normal: The stored procedure runs where the transformation exists in the mapping on a row-by-row basis. This is useful for calling the stored procedure for each row of data that passes through the mapping, such as running a calculation against an input port. Connected stored procedures run only in normal mode.
* Pre-load of the Source: Before the session retrieves data from the source, the stored procedure runs. This is useful for verifying the existence of tables or performing joins of data in a temporary table.
* Post-load of the Source: After the session retrieves data from the source, the stored procedure runs. This is useful for removing temporary tables.
* Pre-load of the Target: Before the session sends data to the target, the stored procedure runs. This is useful for verifying target tables or disk space on the target system.
* Post-load of the Target: After the session sends data to the target, the stored procedure runs. This is useful for re-creating indexes on the database.

A connected stored procedure transformation runs only in Normal mode. A unconnected stored procedure transformation runs in all the above modes.  
  
**8.** What is execution order in stored procedure transformation?  
  
The order in which the Integration Service calls the stored procedure used in the transformation, relative to any other stored procedures in the same mapping. Only used when the Stored Procedure Type is set to anything except Normal and more than one stored procedure exists.  
  
**9.** What is PROC\_RESULT in stored procedure transformation?  
  
PROC\_RESULT is a system variable, where the output of an unconnected stored procedure transformation is assigned by default.  
  
**10.** What are the parameter types in a stored procedure?  
  
There are three types of parameters exist in a stored procedure:

* IN: Input passed to the stored procedure
* OUT: Output returned from the stored procedure
* INOUT: Defines the parameter as both input and output. Only Oracle supports this parameter type.

# Sorter Transformation

**1.** What is a sorter transformation?  
  
Sorter transformation is used to sort the data. You can sort the data either in ascending or descending order according to a specified sort key.  
  
**2.** Why sorter is an active transformation?  
  
As sorter transformation can suppress the duplicate records in the source, it is called an active transformation.  
  
**3.** How to improve the performance of a session using sorter transformation?  
  
Sort the data using sorter transformation before passing in to aggregator or joiner transformation. As the data is sorted, the integration service uses the memory to do aggregate and join operations and does not use cache files to process the data.

**Update Strategy Transformation**

**1.** What is an update strategy transformation?  
  
Update strategy transformation is used to flag source rows for insert, update, delete or reject within a mapping. Based on this flagging each row will be either inserted or updated or deleted from the target. Alternatively the row can be rejected.  
  
**2.** Why update strategy is an active transformation?  
  
As update strategy transformation can reject rows, it is called as an active transformation.  
  
**3.** What are the constants used in update strategy transformation for flagging the rows?

* DD\_INSERT is used for inserting the rows. The numeric value is 0.
* DD\_UPDATE is used for updating the rows. The numeric value is 1.
* DD\_DELETE is used for deleting the rows. The numeric value is 2.
* DD\_REJECT is used for rejecting the rows. The numeric value is 3.

**4.** If you place an aggregator after the update strategy transformation, how the output of aggregator will be affected?  
  
The update strategy transformation flags the rows for insert, update and delete of reject before you perform aggregate calculation. How you flag a particular row determines how the aggregator transformation treats any values in that row used in the calculation. For example, if you flag a row for delete and then later use the row to calculate the sum, the integration service subtracts the value appearing in this row. If the row had been flagged for insert, the integration service would add its value to the sum.  
  
**5.** How to update the target table without using update strategy transformation?  
  
In the session properties, there is an option 'Treat Source Rows As'. Using this option you can specify whether all the source rows need to be inserted, updated or deleted.  
  
**6.** If you have an update strategy transformation in the mapping, what should be the value selected for 'Treat Source Rows As' option in session properties?  
  
The value selected for the option is 'Data Driven'. The integration service follows the instructions coded in the update strategy transformation.  
  
**7.** If you have an update strategy transformation in the mapping and you did not selected the value 'Data Driven' for 'Treat Source Rows As' option in session, then how the session will behave?  
  
If you do not choose Data Driven when a mapping contains an Update Strategy or Custom transformation, the Workflow Manager displays a warning. When you run the session, the Integration Service does not follow instructions in the Update Strategy transformation in the mapping to determine how to flag rows.  
  
**8.** In which files the data rejected by update strategy transformation will be written?  
  
If the update strategy transformation is configured to Forward Rejected Rows then the integration service forwards the rejected rows to next transformation and writes them to the session reject file. If you do not select the forward reject rows option, the integration service drops rejected rows and writes them to the session log file. If you enable row error handling, the Integration Service writes the rejected rows and the dropped rows to the row error logs. It does not generate a reject file.

**Union Transformation**

**1.** What is a union transformation?  
  
A union transformation is used merge data from multiple sources similar to the UNION ALL SQL statement to combine the results from two or more SQL statements.

**2.** As union transformation gives UNION ALL output, how you will get the UNION output?  
  
Pass the output of union transformation to a sorter transformation. In the properties of sorter transformation check the option select distinct. Alternatively you can pass the output of union transformation to aggregator transformation and in the aggregator transformation specify all ports as group by ports.

**3.** What are the guidelines to be followed while using union transformation?  
  
The following rules and guidelines need to be taken care while working with union transformation:

* You can create multiple input groups, but only one output group.
* All input groups and the output group must have matching ports. The precision, datatype, and scale must be identical across all groups.
* The Union transformation does not remove duplicate rows. To remove duplicate rows, you must add another transformation such as a Router or Filter transformation.
* You cannot use a Sequence Generator or Update Strategy transformation upstream from a Union transformation.
* The Union transformation does not generate transactions.

**4.** Why union transformation is an active transformation?  
  
Union is an active transformation because it combines two or more data streams into one. Though the total number of rows passing into the Union is the same as the total number of rows passing out of it, and the sequence of rows from any given input stream is preserved in the output, the positions of the rows are not preserved, i.e. row number 1 from input stream 1 might not be row number 1 in the output stream. Union does not even guarantee that the output is repeatable

**Sequence Generator Transformation**

**1.** What is a sequence generator transformation?  
  
A Sequence generator transformation generates numeric values. Sequence generator transformation is a passive transformation.  
  
**2.** What is the use of a sequence generator transformation?  
  
A sequence generator is used to create unique primary key values, replace missing primary key values or cycle through a sequential range of numbers.  
  
**3.** What are the ports in sequence generator transformation?  
  
A sequence generator contains two output ports. They are CURRVAL and NEXTVAL.  
  
**4.** What is the maximum number of sequence that a sequence generator can generate?  
  
The maximum value is 9,223,372,036,854,775,807  
  
**5.** When you connect both the NEXTVAL and CURRVAL ports to a target, what will be the output values of these ports?  
  
The output values are  
NEXTVAL  CURRVAL  
1        2  
2        3  
3        4  
4        5  
5        6  
  
**6.** What will be the output value, if you connect only CURRVAL to the target without connecting NEXTVAL?  
  
The integration service passes a constant value for each row.  
  
**7.** What will be the value of CURRVAL in a sequence generator transformation?  
  
CURRVAL is the sum of "NEXTVAL" and "Increment By" Value.  
  
**8.** What is the number of cached values set to default for a sequence generator transformation?  
  
For non-reusable sequence generators, the number of cached values is set to zero.  
For reusable sequence generators, the number of cached values is set to 1000.  
  
**9.** How do you configure a sequence generator transformation?  
  
The following properties need to be configured for a sequence generator transformation:

* Start Value
* Increment By
* End Value
* Current Value
* Cycle
* Number of Cached Values

**List of Informatica Transformation Category Wise**  
  
**Active Transformations      Passive Transformations**      **Both Active &  Passive Transformation**  
1. Aggregator[Aggregator Transformation Icon](http://informatica-tutorials-for-beginners.blogspot.in/2013/11/aggregator-transformation.html)                                1. Data Masking                                 1. Custom[https://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s320/image003.png](http://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s1600/image003.png)  
2. Application Source Qualifier&[https://4.bp.blogspot.com/-qvwNeY8_9zs/UpCGLyAbgcI/AAAAAAAAAKo/SA4XFPCFiXk/s1600/image005.png](http://4.bp.blogspot.com/-qvwNeY8_9zs/UpCGLyAbgcI/AAAAAAAAAKo/SA4XFPCFiXk/s1600/image005.png)    2. Expression[https://3.bp.blogspot.com/-6z4N9TaTKVw/UpCGMeDslFI/AAAAAAAAAK0/rVy-99fas80/s320/image007.png](http://3.bp.blogspot.com/-6z4N9TaTKVw/UpCGMeDslFI/AAAAAAAAAK0/rVy-99fas80/s1600/image007.png)                               2. Lookup[https://1.bp.blogspot.com/-znITgftobFA/UpCGMngr1oI/AAAAAAAAALM/f5dDew4G_9o/s320/image009.png](http://1.bp.blogspot.com/-znITgftobFA/UpCGMngr1oI/AAAAAAAAALM/f5dDew4G_9o/s1600/image009.png)  
3. Source Qualifier[https://3.bp.blogspot.com/-QKETL1GQI6Q/UpCGMwbfb8I/AAAAAAAAALI/0vhjnBa9Z90/s1600/image011.png](http://3.bp.blogspot.com/-QKETL1GQI6Q/UpCGMwbfb8I/AAAAAAAAALI/0vhjnBa9Z90/s1600/image011.png)                       3. External Procedure                          3. Unstructured Data[https://3.bp.blogspot.com/-WYRWClhbQPg/UpCGNDvfCmI/AAAAAAAAALE/QBJBD4g34nU/s320/image013.png](http://3.bp.blogspot.com/-WYRWClhbQPg/UpCGNDvfCmI/AAAAAAAAALE/QBJBD4g34nU/s1600/image013.png)  
4. Sorter[https://4.bp.blogspot.com/-5s8Wy_Xep-4/UpCGNgaObxI/AAAAAAAAALU/SUWZoUVug18/s320/image015.png](http://4.bp.blogspot.com/-5s8Wy_Xep-4/UpCGNgaObxI/AAAAAAAAALU/SUWZoUVug18/s1600/image015.png)                                       4. Input                                              4. SQL[https://3.bp.blogspot.com/-vngp1VJfvOU/UpCGN-gx3rI/AAAAAAAAAMA/r5I5V0mNvNE/s320/image017.png](http://3.bp.blogspot.com/-vngp1VJfvOU/UpCGN-gx3rI/AAAAAAAAAMA/r5I5V0mNvNE/s1600/image017.png)  
5. Union[https://3.bp.blogspot.com/-fslNzWCZlUY/UpCGOA5zNhI/AAAAAAAAALk/7RUN7dSdoXo/s320/image019.png](http://3.bp.blogspot.com/-fslNzWCZlUY/UpCGOA5zNhI/AAAAAAAAALk/7RUN7dSdoXo/s1600/image019.png)                                        5. Output                                           5. Java[https://4.bp.blogspot.com/-YS4dsDh2Y-o/UpCGOgK6vAI/AAAAAAAAALo/EF8a5R3GhJ4/s320/image021.png](http://4.bp.blogspot.com/-YS4dsDh2Y-o/UpCGOgK6vAI/AAAAAAAAALo/EF8a5R3GhJ4/s1600/image021.png)  
6. Update Strategy[https://4.bp.blogspot.com/-OqvaMEq4H-w/UpCGPNPm8kI/AAAAAAAAAL4/L6lI_HMA8Ag/s320/image023.png](http://4.bp.blogspot.com/-OqvaMEq4H-w/UpCGPNPm8kI/AAAAAAAAAL4/L6lI_HMA8Ag/s1600/image023.png)                         6. Sequence Generator[https://2.bp.blogspot.com/-62aS5ECYdto/UpCGPFRvzdI/AAAAAAAAALw/d13RpFxtpzA/s320/image025.png](http://2.bp.blogspot.com/-62aS5ECYdto/UpCGPFRvzdI/AAAAAAAAALw/d13RpFxtpzA/s1600/image025.png)  
7. Rank[https://3.bp.blogspot.com/-ySslVBOWTIk/UpCGP-W8qcI/AAAAAAAAAMI/LsnpPjLfQrk/s320/image027.png](http://3.bp.blogspot.com/-ySslVBOWTIk/UpCGP-W8qcI/AAAAAAAAAMI/LsnpPjLfQrk/s1600/image027.png)                                        7. Stored Procedure[https://3.bp.blogspot.com/-EB6KofsekyY/UpCGQPUbG9I/AAAAAAAAAMU/yWVXCQdu6bw/s320/image029.png](http://3.bp.blogspot.com/-EB6KofsekyY/UpCGQPUbG9I/AAAAAAAAAMU/yWVXCQdu6bw/s1600/image029.png)  
8. Router[https://3.bp.blogspot.com/-LAE27Nt-Wnk/UpCGQRwEysI/AAAAAAAAAMY/77HIv-vdKC0/s320/image031.png](http://3.bp.blogspot.com/-LAE27Nt-Wnk/UpCGQRwEysI/AAAAAAAAAMY/77HIv-vdKC0/s1600/image031.png)                                      8. HTTP[https://3.bp.blogspot.com/-fPbSm2QuYZQ/UpCGQrKv8VI/AAAAAAAAAMc/eLhMQZMJ7yY/s320/image033.png](http://3.bp.blogspot.com/-fPbSm2QuYZQ/UpCGQrKv8VI/AAAAAAAAAMc/eLhMQZMJ7yY/s1600/image033.png)  
9. Filter[https://3.bp.blogspot.com/-ufvQvDdhKr4/UpCGRD8rt_I/AAAAAAAAAMk/5I-iPG8LXsE/s320/image035.png](http://3.bp.blogspot.com/-ufvQvDdhKr4/UpCGRD8rt_I/AAAAAAAAAMk/5I-iPG8LXsE/s1600/image035.png)   
10. Normalizer[https://4.bp.blogspot.com/-e8FSell_gG4/UpCGRTlmMpI/AAAAAAAAAM0/fRAUTMaa080/s320/image037.png](http://4.bp.blogspot.com/-e8FSell_gG4/UpCGRTlmMpI/AAAAAAAAAM0/fRAUTMaa080/s1600/image037.png)   
11. Transaction Control[https://1.bp.blogspot.com/-8zuec1l72Sk/UpCGRf5SWCI/AAAAAAAAAM4/j4yV2HG-IZs/s320/image039.png](http://1.bp.blogspot.com/-8zuec1l72Sk/UpCGRf5SWCI/AAAAAAAAAM4/j4yV2HG-IZs/s1600/image039.png)   
12. Joiner[https://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s320/image041.png](http://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s1600/image041.png)   
13. XML Source Qualifier[https://1.bp.blogspot.com/-L3EZN2AsIuY/UpCGTI-0QDI/AAAAAAAAANY/BGveFL9PVVk/s320/image043.png](http://1.bp.blogspot.com/-L3EZN2AsIuY/UpCGTI-0QDI/AAAAAAAAANY/BGveFL9PVVk/s1600/image043.png)   
14. XML Generator[https://1.bp.blogspot.com/-Be4HSbpGAng/UpCDhadcH6I/AAAAAAAAAKQ/N7AiAZLCIwo/s1600/image045.png](http://1.bp.blogspot.com/-Be4HSbpGAng/UpCDhadcH6I/AAAAAAAAAKQ/N7AiAZLCIwo/s1600/image045.png)  
15. XML Parser[https://2.bp.blogspot.com/-iZ8dXyHCZ00/UpCDhie8urI/AAAAAAAAAKY/drU8ACToBo4/s1600/image047.png](http://2.bp.blogspot.com/-iZ8dXyHCZ00/UpCDhie8urI/AAAAAAAAAKY/drU8ACToBo4/s1600/image047.png)  
  
**Connected and Unconnected Transformation (Both)**              **Connected Transformation**  
1. External Procedure                                                                Rest All transformations are connected.  
2. Stored Procedure[https://3.bp.blogspot.com/-EB6KofsekyY/UpCGQPUbG9I/AAAAAAAAAMU/yWVXCQdu6bw/s320/image029.png](http://3.bp.blogspot.com/-EB6KofsekyY/UpCGQPUbG9I/AAAAAAAAAMU/yWVXCQdu6bw/s1600/image029.png)  
3. Lookup [https://1.bp.blogspot.com/-znITgftobFA/UpCGMngr1oI/AAAAAAAAALM/f5dDew4G_9o/s320/image009.png](http://1.bp.blogspot.com/-znITgftobFA/UpCGMngr1oI/AAAAAAAAALM/f5dDew4G_9o/s1600/image009.png)  
  
**MultiGroup Transformations**  
1. Union[https://3.bp.blogspot.com/-fslNzWCZlUY/UpCGOA5zNhI/AAAAAAAAALk/7RUN7dSdoXo/s320/image019.png](http://3.bp.blogspot.com/-fslNzWCZlUY/UpCGOA5zNhI/AAAAAAAAALk/7RUN7dSdoXo/s1600/image019.png)  
2. Router[https://3.bp.blogspot.com/-LAE27Nt-Wnk/UpCGQRwEysI/AAAAAAAAAMY/77HIv-vdKC0/s320/image031.png](http://3.bp.blogspot.com/-LAE27Nt-Wnk/UpCGQRwEysI/AAAAAAAAAMY/77HIv-vdKC0/s1600/image031.png)  
3. Joiner[https://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s320/image041.png](http://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s1600/image041.png)  
4. Custom[https://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s320/image003.png](http://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s1600/image003.png)  
5. Unstructured Data[https://3.bp.blogspot.com/-WYRWClhbQPg/UpCGNDvfCmI/AAAAAAAAALE/QBJBD4g34nU/s320/image013.png](http://3.bp.blogspot.com/-WYRWClhbQPg/UpCGNDvfCmI/AAAAAAAAALE/QBJBD4g34nU/s1600/image013.png)   
6. XML Source Qualifier[https://1.bp.blogspot.com/-L3EZN2AsIuY/UpCGTI-0QDI/AAAAAAAAANY/BGveFL9PVVk/s320/image043.png](http://1.bp.blogspot.com/-L3EZN2AsIuY/UpCGTI-0QDI/AAAAAAAAANY/BGveFL9PVVk/s1600/image043.png)   
7. XML Generator[https://1.bp.blogspot.com/-Be4HSbpGAng/UpCDhadcH6I/AAAAAAAAAKQ/N7AiAZLCIwo/s1600/image045.png](http://1.bp.blogspot.com/-Be4HSbpGAng/UpCDhadcH6I/AAAAAAAAAKQ/N7AiAZLCIwo/s1600/image045.png)  
8. XML Parser[https://2.bp.blogspot.com/-iZ8dXyHCZ00/UpCDhie8urI/AAAAAAAAAKY/drU8ACToBo4/s1600/image047.png](http://2.bp.blogspot.com/-iZ8dXyHCZ00/UpCDhie8urI/AAAAAAAAAKY/drU8ACToBo4/s1600/image047.png)  
9. XML Target Definition  
  
**Blocking Transformations**   
1.Custom transformation with Input may block property enabled.[https://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s320/image003.png](http://2.bp.blogspot.com/-GG5WRDdiXFE/UpCGL6fsKKI/AAAAAAAAAKs/_SVKYdE67z8/s1600/image003.png)  
2. Joiner transformation configured for unsorted Input.[https://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s320/image041.png](http://3.bp.blogspot.com/-NJZ1PLJWCoY/UpCGSHMG24I/AAAAAAAAANA/l-oK4qzroXg/s1600/image041.png)

### Difference between Sorter and Rank transformation in Informatica

|  |  |
| --- | --- |
| Sorter | Rank |
| Sorter is used to Sort the data either ASC or DSC | Rank is used to arrange data from top or bottom Group by can be done using Rank |
| Sorter can be used to remove duplicates(Use Distinct ouptut) | Using Rank we cannot remove duplicates |
| In sorter we cannot assign values to the ports | In rank we can assign variables and write non-aggregate expressions also. |
| In sorter the Integration Service uses Sorter Cache to perform the sort operation. | In Rank the Integration Service stores group information in an index cache and row data in a data cache. |

### What are the differences between Connected and Unconnected Lookup?

The differences are illustrated in the below table:

| **Connected Lookup** | **Unconnected Lookup** |
| --- | --- |
| Connected lookup participates in dataflow and receives input directly from the pipeline | Unconnected lookup receives input values from the result of a LKP: expression in another transformation |
| Connected lookup can use both dynamic and static cache | Unconnected Lookup cache can NOT be dynamic |
| Connected lookup can return more than one column value ( output port ) | Unconnected Lookup can return only one column value i.e. output port |
| Connected lookup caches all lookup columns | Unconnected lookup caches only the lookup output ports in the lookup conditions and the return port |
| Supports user-defined default values (i.e. value to return when lookup conditions are not satisfied) | Does not support user defined default values |

### What is meant by active and passive transformation?

An active transformation is the one that performs any of the following actions:

1. Change the number of rows between transformation input and output. Example: Filter transformation
2. Change the transaction boundary by defining commit or rollback points., example transaction control transformation
3. Change the row type, example Update strategy is active because it flags the rows for insert, delete, update or reject

On the other hand a passive transformation is the one which does not change the number of rows that pass through it. Example: Expression transformation.

### What is the difference between Router and Filter?

Following differences can be noted:

| **Router** | **Filter** |
| --- | --- |
| Router transformation divides the incoming records into multiple groups based on some condition. Such groups can be mutually inclusive (Different groups may contain same record) | Filter transformation restricts or blocks the incoming record set based on one given condition. |
| Router transformation itself does not block any record. If a certain record does not match any of the routing conditions, the record is routed to default group | Filter transformation does not have a default group. If one record does not match filter condition, the record is blocked |
| Router acts like CASE.. WHEN statement in SQL (Or Switch().. Case statement in C) | Filter acts like WHERE condition is SQL. |

Do you want to study from the biggest collection of Informatica questions and answers? Where each question is hand-picked from real-life Interviews and answers prepared by the industry experts? Do you want to download the question-answer set in PDF format for offline study? If yes, get the Master Informatica Question Answer Set.   
  
Get It Now

### What can we do to improve the performance of Informatica Aggregator Transformation?

Aggregator performance improves dramatically if records are sorted before passing to the aggregator and "sorted input" option under aggregator properties is checked. The record set should be sorted on those columns that are used in Group By operation.

It is often a good idea to sort the record set in database level ([click here to see why?](https://dwbi.org/etl/informatica/154-informatica-oracle-sort-performance-test)) e.g. inside a source qualifier transformation, unless there is a chance that already sorted records from source qualifier can again become unsorted before reaching aggregator

You may also read [this article](https://dwbi.org/etl/informatica/160-tuning-informatica-aggregator) to know how to tune the performance of aggregator transformation

### What are the different lookup cache(s)?

Informatica Lookups can be cached or un-cached (No cache). And Cached lookup can be either static or dynamic. A **static cache** is one which does not modify the cache once it is built and it remains same during the session run. On the other hand, A [**dynamic cache**](https://dwbi.org/etl/informatica/138-dynamic-lookup-cache)is refreshed during the session run by inserting or updating the records in cache based on the incoming source data. By default, Informatica cache is static cache.

A lookup cache can also be divided as [**persistent** or **non-persistent**](https://dwbi.org/etl/informatica/136-implementing-informatica-persistent-cache)based on whether Informatica retains the cache even after the completion of session run or deletes it.

### How can we update a record in target table without using Update strategy?

A target table can be updated without using 'Update Strategy'. For this, we need to define the key in the target table in Informatica level and then we need to connect the key and the field we want to update in the mapping Target. In the session level, we should set the target property as "Update as Update" and check the "Update" check-box.

Let's assume we have a target table "Customer" with fields as "Customer ID", "Customer Name" and "Customer Address". Suppose we want to update "Customer Address" without an Update Strategy. Then we have to define "Customer ID" as primary key in Informatica level and we will have to connect Customer ID and Customer Address fields in the mapping. If the session properties are set correctly as described above, then the mapping will only update the customer address field for all matching customer IDs.

### Under what condition selecting Sorted Input in aggregator may fail the session?

1. If the input data is not sorted correctly, the session will fail.
2. Also if the input data is properly sorted, the session may fail if the sort order by ports and the group by ports of the aggregator are not in the same order.

### Why is Sorter an Active Transformation?

This is because we can select the "distinct" option in the sorter property.

When the Sorter transformation is configured to treat output rows as distinct, it assigns all ports as part of the sort key. The Integration Service discards duplicate rows compared during the sort operation. The number of Input Rows will vary as compared with the Output rows and hence it is an Active transformation.

### Is lookup an active or passive transformation?

From Informatica 9x, Lookup transformation can be configured as as "Active" transformation.

Find out [How to configure lookup as active transformation](https://dwbi.org/etl/informatica/139-active-lookup-transformation)

However, in the older versions of Informatica, lookup used to be a passive transformation

### What is the difference between Static and Dynamic Lookup Cache?

We can configure a Lookup transformation to cache the underlying lookup table. In case of static or read-only lookup cache the Integration Service caches the lookup table at the beginning of the session and does not update the lookup cache while it processes the Lookup transformation.

In case of dynamic lookup cache the Integration Service dynamically inserts or updates data in the lookup cache and passes the data to the target. The dynamic cache is synchronized with the target.

In case you are wondering why do we need to make lookup cache dynamic, read this article on[dynamic lookup](https://dwbi.org/etl/informatica/138-dynamic-lookup-cache)

### What is the difference between STOP and ABORT options in Workflow Monitor?

When we issue the STOP command on the executing session task, the Integration Service stops reading data from source. It continues processing, writing and committing the data to targets. If the Integration Service cannot finish processing and committing data, we can issue the abort command.

In contrast ABORT command has a timeout period of 60 seconds. If the Integration Service cannot finish processing and committing data within the timeout period, it kills the DTM process and terminates the session.

### What are the new features of Informatica 9.x in developer level?

From a developer's perspective, some of the new features in Informatica 9.x are as follows:

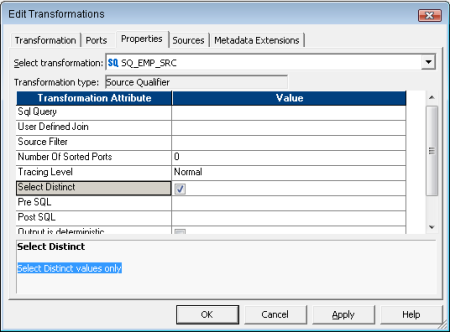
* Now Lookup can be configured as an active transformation - it can return multiple rows on successful match
* Now you can write SQL override on un-cached lookup also. Previously you could do it only on cached lookup
* You can control the size of your session log. In a real-time environment you can control the session log file size or time
* Database deadlock resilience feature - this will ensure that your session does not immediately fail if it encounters any database deadlock, it will now retry the operation again. You can configure number of retry attempts.

### How to Delete duplicate row using Informatica

#### Scenario 1: Duplicate rows are present in relational database

Suppose we have Duplicate records in Source System and we want to load only the unique records in the Target System eliminating the duplicate rows. What will be the approach?

Assuming that the source system is a **Relational Database**, to eliminate duplicate records, we can check the **Distinct** option of the **Source Qualifier** of the source table and load the target accordingly.



But what if the source is a flat file? Then how can we remove the duplicates from flat file source?

#### Scenario 2: Deleting duplicate rows / selecting distinct rows for FLAT FILE sources

Here since the source system is a **Flat File** you will not be able to select the distinct option in the source qualifier as it will be disabled due to flat file source table. Hence the next approach may be we use a**Sorter Transformation** and check the **Distinct** option. When we select the distinct option all the columns will the selected as keys, in ascending order by default.

#### Deleting Duplicate Record Using Informatica Aggregator

Other ways to handle duplicate records in source batch run is to use an **Aggregator Transformation** and using the **Group By** checkbox on the ports having duplicate occurring data. Here you can have the flexibility to select the *last or the first* of the duplicate column value records.

There is yet another option to ensure duplicate records are not inserted in the target. That is through**Dynamic lookup** cache. Using Dynamic Lookup Cache of the target table and associating the input ports with the lookup port and checking the Insert Else Update option will help to eliminate the duplicate records in source and hence loading unique records in the target.   
  
For more details check, [Dynamic Lookup Cache](https://dwbi.org/etl/informatica/138-dynamic-lookup-cache)

### Loading Multiple Target Tables Based on Conditions

#### Scenario

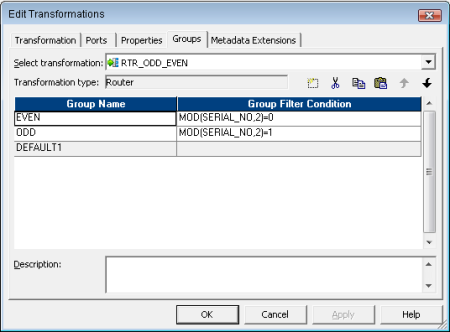
Suppose we have some serial numbers in a flat file source. We want to load the serial numbers in two target files one containing the EVEN serial numbers and the other file having the ODD ones.

#### Answer

After the Source Qualifier place a **Router Transformation**. Create two **Groups** namely **EVEN and ODD**, with filter conditions as:

MOD(SERIAL\_NO,2)=0 and MOD(SERIAL\_NO,2)=1

... respectively. Then output the two groups into two flat file targets.



### Normalizer Related Questions

#### Scenario 1

Suppose in our Source Table we have data as given below:

| **Student Name** | **Maths** | **Life Science** | **Physical Science** |
| --- | --- | --- | --- |
| Sam | 100 | 70 | 80 |
| John | 75 | 100 | 85 |
| Tom | 80 | 100 | 85 |

We want to load our Target Table as:

| **Student Name** | **Subject Name** | **Marks** |
| --- | --- | --- |
| Sam | Maths | 100 |
| Sam | Life Science | 70 |
| Sam | Physical Science | 80 |
| John | Maths | 75 |
| John | Life Science | 100 |
| John | Physical Science | 85 |
| Tom | Maths | 80 |
| Tom | Life Science | 100 |
| Tom | Physical Science | 85 |

Describe your approach.

#### Answer

Here to convert the Rows to Columns we have to use the **Normalizer Transformation** followed by an Expression Transformation to Decode the column taken into consideration. For more details on how the mapping is performed please visit [Working with Normalizer](https://dwbi.org/etl/informatica/147-using-informatica-normalizer-transformation)

#### Question

Name the transformations which converts one to many rows i.e increases the i/p:o/p row count. Also what is the name of its reverse transformation.

#### Answer

**Normalizer** as well as **Router** Transformations are the Active transformation which can increase the number of input rows to output rows.

**Aggregator** Transformation performs the reverse action of Normalizer transformation.

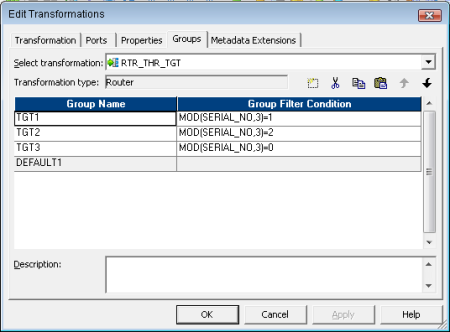
#### Scenario 2

Suppose we have a source table and we want to load three target tables based on source rows such that first row moves to first target table, second row in second target table, third row in third target table, fourth row again in first target table so on and so forth. Describe your approach.

#### Answer

We can clearly understand that we need a **Router transformation** to route or filter source data to the three target tables. Now the question is what will be the filter conditions. First of all we need an**Expression Transformation** where we have all the source table columns and along with that we have another i/o port say seq\_num, which is gets sequence numbers for each source row from the port **NextVal**of a **Sequence Generator start value 0 and increment by 1**. Now the filter condition for the three router groups will be:

* MOD(SEQ\_NUM,3)=1 connected to 1st target table
* MOD(SEQ\_NUM,3)=2 connected to 2nd target table
* MOD(SEQ\_NUM,3)=0 connected to 3rd target table



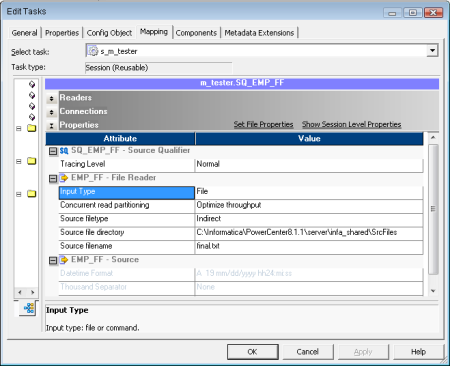
### Loading Multiple Flat Files using one mapping

#### Scenario

Suppose we have ten source flat files of same structure. How can we load all the files in target database in a single batch run using a single mapping.

#### Answer

After we create a mapping to load data in target database from flat files, next we move on to the session property of the Source Qualifier. To load a set of source files we need to create a file say final.txt containing the source falt file names, ten files in our case and set the **Source filetype** option as **Indirect**. Next point this flat file final.txt fully qualified through **Source file directory** and **Source filename**.



### Aggregator Transformation Related Questions

#### How can we implement Aggregation operation without using an Aggregator Transformation in Informatica?

#### Answer

We will use the very basic concept of the **Expression Transformation** that at a time we can access the previous row data as well as the currently processed data in an expression transformation. What we need is simple Sorter, Expression and Filter transformation to achieve aggregation at Informatica level.

For detailed understanding visit [Aggregation without Aggregator](https://dwbi.org/etl/informatica/135-aggregation-with-out-informatica-aggregator)

#### Scenario

Suppose in our Source Table we have data as given below:

| **Student Name** | **Subject Name** | **Marks** |
| --- | --- | --- |
| Sam | Maths | 100 |
| Tom | Maths | 80 |
| Sam | Physical Science | 80 |
| John | Maths | 75 |
| Sam | Life Science | 70 |
| John | Life Science | 100 |
| John | Physical Science | 85 |
| Tom | Life Science | 100 |
| Tom | Physical Science | 85 |

We want to load our Target Table as:

| **Student Name** | **Maths** | **Life Science** | **Physical Science** |
| --- | --- | --- | --- |
| Sam | 100 | 70 | 80 |
| John | 75 | 100 | 85 |
| Tom | 80 | 100 | 85 |

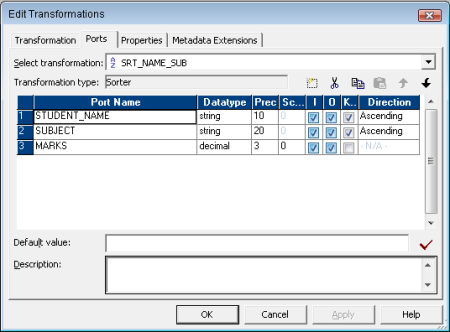
Describe your approach.

#### Answer

Here our scenario is to convert many rows to one rows, and the transformation which will help us to achieve this is **Aggregator**.

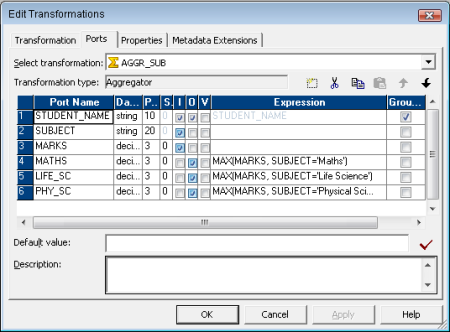
Our Mapping will look like this:

We will sort the source data based on STUDENT\_NAME ascending followed by SUBJECT ascending.



Now based on STUDENT\_NAME in **GROUP BY** clause the following output subject columns are populated as

* MATHS: MAX(MARKS, SUBJECT=Maths)
* LIFE\_SC: MAX(MARKS, SUBJECT=Life Science)
* PHY\_SC: MAX(MARKS, SUBJECT=Physical Science)



### Revisiting Source Qualifier Transformation

#### What is a Source Qualifier? What are the tasks we can perform using a SQ and why it is an ACTIVE transformation?

**Ans.** A **Source Qualifier** is an Active and Connected Informatica transformation that reads the rows from a relational database or flat file source.

* We can configure the **SQ** to **join** [Both **INNER** as well as **OUTER JOIN**] data originating from the same source database.
* We can use a source **filter** to reduce the number of rows the Integration Service queries.
* We can specify a number for **sorted ports** and the Integration Service adds an ORDER BY clause to the default SQL query.
* We can choose **Select Distinct**option for relational databases and the Integration Service adds a SELECT DISTINCT clause to the default SQL query.
* Also we can write **Custom/Used Defined SQL** query which will override the default query in the SQ by changing the default settings of the transformation properties.
* Also we have the option to write **Pre** as well as **Post SQL** statements to be executed before and after the SQ query in the source database.

Since the transformation provides us with the property **Select Distinct**, when the Integration Service adds a SELECT DISTINCT clause to the default SQL query, which in turn affects the number of rows returned by the Database to the Integration Service and hence it is an Active transformation.

#### What happens to a mapping if we alter the datatypes between Source and its corresponding Source Qualifier?

**Ans.** The Source Qualifier transformation displays the transformation datatypes. The transformation datatypes determine how the source database binds data when the Integration Service reads it.

Now if we alter the datatypes in the Source Qualifier transformation or the **datatypes in the source definition and Source Qualifier transformation do not match,** the Designer marks the **mapping as invalid** when we save it.

#### Suppose we have used the Select Distinct and the Number Of Sorted Ports property in the SQ and then we add Custom SQL Query. Explain what will happen.

**Ans.** Whenever we add Custom SQL or SQL override query it **overrides** the User-Defined Join, Source Filter, Number of Sorted Ports, and Select Distinct settings in the Source Qualifier transformation. Hence only the user defined SQL Query will be fired in the database and all the **other options will be ignored** .

#### Describe the situations where we will use the Source Filter, Select Distinct and Number Of Sorted Ports properties of Source Qualifier transformation.

**Ans.** **Source Filter** option is used basically to reduce the number of rows the Integration Service queries so as to improve performance.

**Select Distinct** option is used when we want the Integration Service to select unique values from a source, filtering out unnecessary data earlier in the data flow, which might improve performance.

**Number Of Sorted Ports** option is used when we want the source data to be in a sorted fashion so as to use the same in some following transformations like Aggregator or Joiner, those when configured for sorted input will improve the performance.

#### What will happen if the SELECT list COLUMNS in the Custom override SQL Query and the OUTPUT PORTS order in SQ transformation do not match?

**Ans.** Mismatch or Changing the order of the list of selected columns to that of the connected transformation output ports may result is **session failure.**

#### What happens if in the Source Filter property of SQ transformation we include keyword WHERE say, WHERE CUSTOMERS.CUSTOMER\_ID > 1000.

**Ans.** We use source filter to reduce the number of source records. If we include the string **WHERE** in the source filter, the Integration Service **fails the session**.

#### Describe the scenarios where we go for Joiner transformation instead of Source Qualifier transformation.

**Ans.** While joining Source Data of **heterogeneous sources** as well as to join **flat files** we will use the Joiner transformation. Use the Joiner transformation when we need to join the following types of sources:

* Join data from different Relational Databases.
* Join data from different Flat Files.
* Join relational sources and flat files.

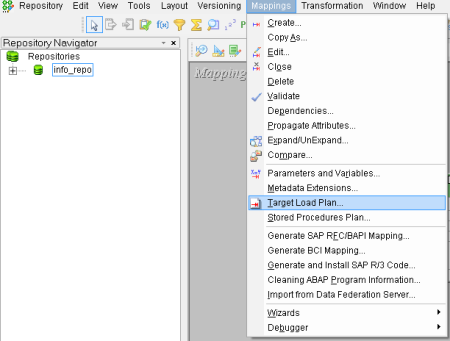
#### What is the maximum number we can use in Number Of Sorted Ports for Sybase source system.

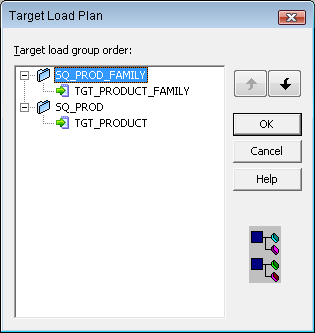
**Ans.** Sybase supports a maximum of **16** columns in an ORDER BY clause. So if the source is Sybase, do not sort more than 16 columns.

#### Suppose we have two Source Qualifier transformations SQ1 and SQ2 connected to Target tables TGT1 and TGT2 respectively. How do you ensure TGT2 is loaded after TGT1?

**Ans.** If we have multiple Source Qualifier transformations connected to multiple targets, we can designate the order in which the Integration Service loads data into the targets.

In the Mapping Designer, We need to configure the **Target Load Plan** based on the Source Qualifier transformations in a mapping to specify the required loading order.

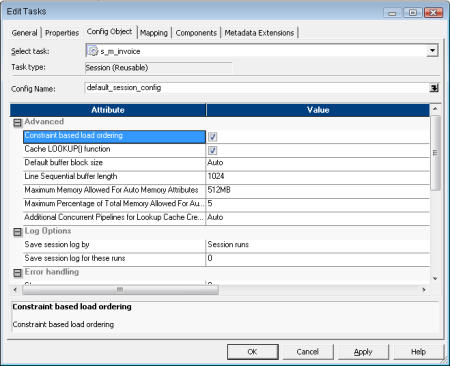




#### Suppose we have a Source Qualifier transformation that populates two target tables. How do you ensure TGT2 is loaded after TGT1?

**Ans.** In the Workflow Manager, we can Configure **Constraint based load ordering** for a session. The Integration Service orders the target load on a row-by-row basis. For every row generated by an active source, the Integration Service loads the corresponding transformed row first to the primary key table, then to the foreign key table.

Hence if we have one Source Qualifier transformation that provides data for multiple target tables having primary and foreign key relationships, we will go for Constraint based load ordering.



### Revisiting Filter Transformation

**Q19.** What is a Filter Transformation and why it is an Active one?

**Ans.** A **Filter** transformation is an **Active** and **Connected** transformation that can filter rows in a mapping.

Only the rows that meet the **Filter Condition** pass through the Filter transformation to the next transformation in the pipeline. TRUE and FALSE are the implicit return values from any filter condition we set. If the filter condition evaluates to NULL, the row is assumed to be FALSE.

The numeric equivalent of FALSE is zero (0) and any non-zero value is the equivalent of TRUE.

As an **ACTIVE** transformation, the Filter transformation may change the number of rows passed through it. A filter condition returns TRUE or FALSE for each row that passes through the transformation, depending on whether a row meets the specified condition. Only rows that return TRUE pass through this transformation. Discarded rows do not appear in the session log or reject files.

**Q20.** What is the difference between Source Qualifier transformations Source Filter to Filter transformation?

**Ans.**

| **SQ Source Filter** | **Filter Transformation** |
| --- | --- |
| Source Qualifier transformation filters rows when read from a source. | Filter transformation filters rows from within a mapping |
| Source Qualifier transformation can only filter rows from Relational Sources. | Filter transformation filters rows coming from any type of source system in the mapping level. |
| Source Qualifier limits the row set extracted from a source. | Filter transformation limits the row set sent to a target. |
| Source Qualifier reduces the number of rows used throughout the mapping and hence it provides better performance. | To maximize session performance, include the Filter transformation as close to the sources in the mapping as possible to filter out unwanted data early in the flow of data from sources to targets. |
| The filter condition in the Source Qualifier transformation only uses standard SQL as it runs in the database. | Filter Transformation can define a condition using any statement or transformation function that returns either a TRUE or FALSE value. |

### Revisiting Joiner Transformation

**Q21.** What is a Joiner Transformation and why it is an Active one?

**Ans.** A **Joiner** is an **Active** and **Connected** transformation used to join source data from the same source system or from two related heterogeneous sources residing in different locations or file systems.

The Joiner transformation joins sources with at least one matching column. The Joiner transformation uses a condition that matches one or more pairs of columns between the two sources.

The two input pipelines include a master pipeline and a detail pipeline or a master and a detail branch. The master pipeline ends at the Joiner transformation, while the detail pipeline continues to the target.

In the Joiner transformation, we must configure the transformation properties namely Join Condition, Join Type and Sorted Input option to improve Integration Service performance.

The join condition contains ports from both input sources that must match for the Integration Service to join two rows. Depending on the type of join selected, the Integration Service either **adds the row to the result set or discards the row**.

The Joiner transformation produces result sets based on the join type, condition, and input data sources. Hence it is an Active transformation.

**Q22.** State the limitations where we cannot use Joiner in the mapping pipeline.

**Ans.** The Joiner transformation accepts input from most transformations. However, following are the limitations:

* Joiner transformation cannot be used when either of the input pipeline contains an **Update Strategy**transformation.
* Joiner transformation cannot be used if we connect a **Sequence Generator** transformation directly before the Joiner transformation.

**Q23.** Out of the two input pipelines of a joiner, which one will you set as the master pipeline?

**Ans.** During a session run, the Integration Service compares **each row of the master source** against the detail source. The master and detail sources need to be configured for **optimal performance**.

To improve performance for an **Unsorted Joiner** transformation, use the source with **fewer rows** as the master source. The fewer unique rows in the master, the fewer iterations of the join comparison occur, which speeds the join process.

When the Integration Service processes an unsorted Joiner transformation, it reads all master rows before it reads the detail rows. The Integration Service blocks the detail source while it **caches rows from the master source**. Once the Integration Service reads and caches all master rows, it unblocks the detail source and reads the detail rows.

To improve performance for a **Sorted Joiner** transformation, use the source with **fewer duplicate key values** as the master source.

When the Integration Service processes a sorted Joiner transformation, it blocks data based on the mapping configuration and it stores **fewer rows** in the cache, increasing performance.

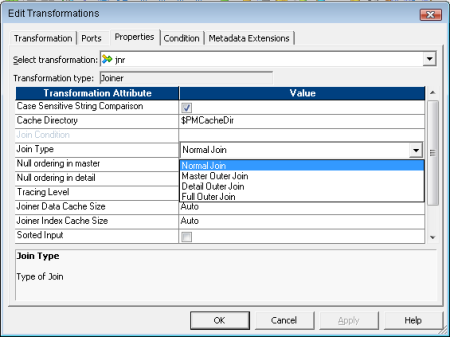
Blocking logic is possible if master and detail input to the Joiner transformation originate from **different sources**. Otherwise, it does not use blocking logic. Instead, it stores**more rows** in the cache.

**Q24.** What are the different types of Joins available in Joiner Transformation?

**Ans.** In SQL, a join is a relational operator that combines data from multiple tables into a single result set. The Joiner transformation is similar to an SQL join except that data can originate from different types of sources.

The Joiner transformation supports the following **types of joins** :

* **Normal**
* **Master Outer**
* **Detail Outer**
* **Full Outer**



**Note:** A **normal or master outer** join **performs faster** than a full outer or detail outer join.

**Q25.** Define the various Join Types of Joiner Transformation.

**Ans.**

* In a **normal join** , the Integration Service discards all rows of data from the master and detail source that do not match, based on the join condition.
* A **master outer join** keeps all rows of data from the detail source and the matching rows from the master source. It discards the unmatched rows from the master source.
* A **detail outer** join keeps all rows of data from the master source and the matching rows from the detail source. It discards the unmatched rows from the detail source.
* A **full outer** join keeps all rows of data from both the master and detail sources.

**Q26.** Describe the impact of number of join conditions and join order in a Joiner Transformation.

**Ans.** We can define **one or more conditions** based on **equality** between the specified master and detail sources. Both ports in a condition must have the **same datatype**.

If we need to use two ports in the join condition with non-matching datatypes we must convert the datatypes so that they match. The Designer validates datatypes in a join condition.

**Additional ports** in the join condition **increases the time** necessary to join two sources.

The order of the ports in the join condition can impact the performance of the Joiner transformation. If we use multiple ports in the join condition, the Integration Service compares the ports in the order we specified.

**NOTE:** Only equality operator is available in joiner join condition.

**Q27.** How does Joiner transformation treat NULL value matching.

**Ans.** The Joiner transformation **does not match null values**.

For example, if both EMP\_ID1 and EMP\_ID2 contain a row with a null value, the Integration Service does not consider them a match and does not join the two rows.

To join rows with null values, replace null input with **default values** in the Ports tab of the joiner, and then join on the default values.

**Note:** If a result set includes fields that do not contain data in either of the sources, the Joiner transformation populates the empty fields with null values. If we know that a field will return a NULL and we do not want to insert NULLs in the target, set a default value on the Ports tab for the corresponding port.

**Q28.** Suppose we configure Sorter transformations in the master and detail pipelines with the following sorted ports in order: ITEM\_NO, ITEM\_NAME, PRICE. When we configure the join condition, what are the guidelines we need to follow to maintain the sort order?

**Ans.** If we have sorted both the master and detail pipelines in order of the ports say ITEM\_NO, ITEM\_NAME and PRICE we must ensure that:

* Use ITEM\_NO in the First Join Condition.
* If we add a Second Join Condition, we must use ITEM\_NAME.
* If we want to use PRICE as a Join Condition apart from ITEM\_NO, we must also use ITEM\_NAME in the Second Join Condition.
* If we skip ITEM\_NAME and join on ITEM\_NO and PRICE, we will **lose the input sort order** and the Integration Service **fails the session**.

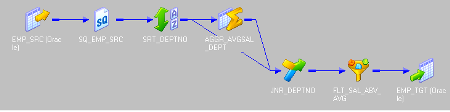
**Q29.** What are the transformations that cannot be placed between the sort origin and the Joiner transformation so that we do not lose the input sort order.

**Ans.** The best option is to place the Joiner transformation directly after the sort origin to maintain sorted data. However do not place any of the following transformations between the sort origin and the Joiner transformation:

* Custom
* **Unsorted**Aggregator
* Normalizer
* Rank
* Union transformation
* XML Parser transformation
* XML Generator transformation
* Mapplet [if it contains any one of the above mentioned transformations]

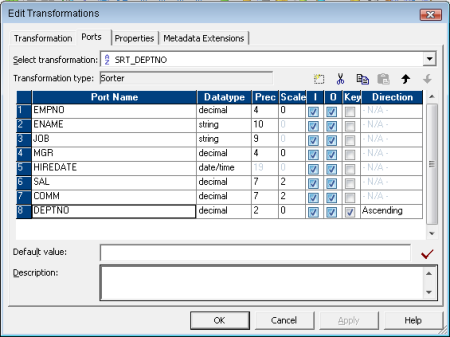
**Q30.** Suppose we have the EMP table as our source. In the target we want to view those employees whose salary is greater than or equal to the average salary for their departments. Describe your mapping approach.

**Ans.** Our Mapping will look like this:



To start with the mapping we need the following transformations:

After the Source qualifier of the EMP table place a **Sorter Transformation** . Sort based on **DEPTNO**port.

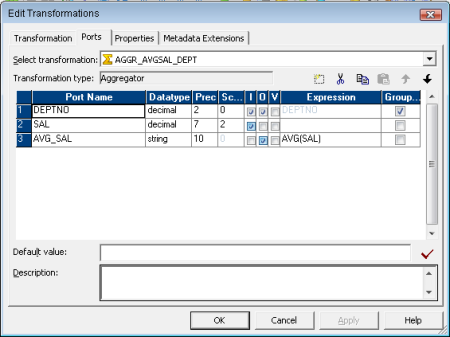


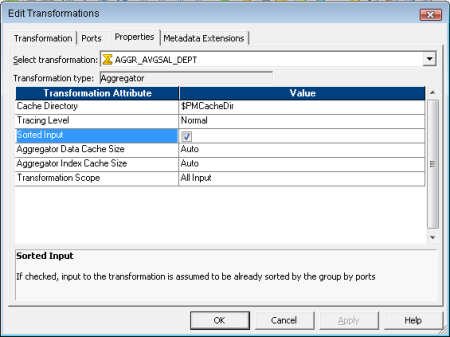
Next we place a **Sorted Aggregator Transformation**. Here we will find out the **AVERAGE SALARY** for each (GROUP BY) **DEPTNO**.

When we perform this aggregation, we lose the data for individual employees.

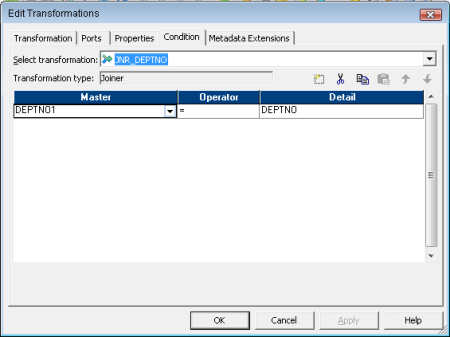
To maintain employee data, we must pass a branch of the pipeline to the Aggregator Transformation and pass a branch with the same sorted source data to the Joiner transformation to maintain the original data.

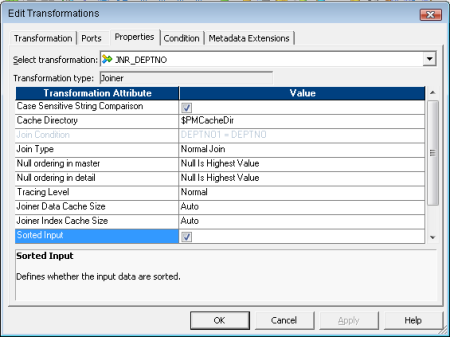
When we join both branches of the pipeline, we join the aggregated data with the original data.





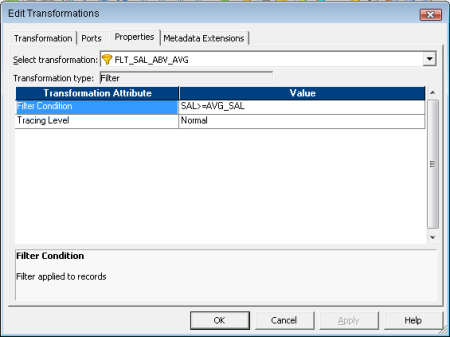
So next we need **Sorted Joiner Transformation** to join the sorted aggregated data with the original data, based on **DEPTNO**. Here we will be taking the aggregated pipeline as the Master and original dataflow as Detail Pipeline.





After that we need a **Filter Transformation** to filter out the employees having salary less than average salary for their department.

Filter Condition: **SAL>=AVG\_SAL**



Lastly we have the Target table instance.

### Revisiting Sequence Generator Transformation

**Q31.** What is a Sequence Generator Transformation?

**Ans.** A **Sequence Generator** transformation is a **Passive** and **Connected** transformation that generates numeric values. It is used to create unique primary key values, replace missing primary keys, or cycle through a sequential range of numbers. This transformation by **default** contains **ONLY Two OUTPUT** ports namely **CURRVAL** and **NEXTVAL**. We cannot edit or delete these ports neither we cannot add ports to this unique transformation. We can create approximately two billion unique numeric values with the widest range from 1 to 2147483647.

**Q32.** Define the Properties available in Sequence Generator transformation in brief.

**Ans.** Sequence Generator:

| **Properties** | **Description** |
| --- | --- |
| Start Value | Start value of the generated sequence that we want the Integration Service to use if we use the Cycle option. If we select Cycle, the Integration Service cycles back to this value when it reaches the end value. Default is 0. |
| Increment By | Difference between two consecutive values from the NEXTVAL port.Default is 1. |
| End Value | Maximum value generated by SeqGen. After reaching this value the session will fail if the sequence generator is not configured to cycle.Default is 2147483647. |
| Current Value | Current value of the sequence. Enter the value we want the Integration Service to use as the first value in the sequence. Default is 1. |
| Cycle | If selected, when the Integration Service reaches the configured end value for the sequence, it wraps around and starts the cycle again, beginning with the configured Start Value. |
| Number of Cached Values | Number of sequential values the Integration Service caches at a time. Default value for a standard Sequence Generator is 0. Default value for a reusable Sequence Generator is 1,000. |
| Reset | Restarts the sequence at the current value each time a session runs.This option is disabled for reusable Sequence Generator transformations. |

**Q33.** Suppose we have a source table populating two target tables. We connect the NEXTVAL port of the Sequence Generator to the surrogate keys of both the target tables.

Will the Surrogate keys in both the target tables be same? If not how can we flow the same sequence values in both of them.

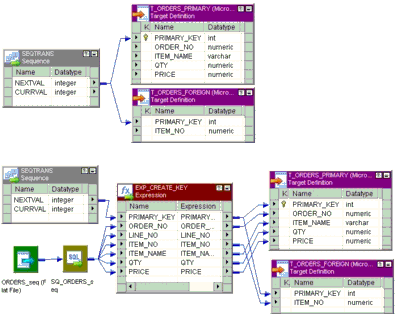
**Ans.** When we connect the **NEXTVAL** output port of the **Sequence Generator** directly to the surrogate key columns of the target tables, the **Sequence number will not be the same**.

A block of sequence numbers is sent to one target tables surrogate key column. The second targets receives a block of sequence numbers from the Sequence Generator transformation only after the first target table receives the block of sequence numbers.

Suppose we have 5 rows coming from the source, so the targets will have the sequence values as TGT1 (1,2,3,4,5) and TGT2 (6,7,8,9,10). [Taken into consideration Start Value 0, Current value 1 and Increment by 1.

Now suppose the requirement is like that we need to have the same surrogate keys in both the targets.

Then the easiest way to handle the situation is to put an **Expression Transformation** in between the Sequence Generator and the Target tables. The SeqGen will pass unique values to the expression transformation, and then the rows are routed from the expression transformation to the targets.



**Q34.** Suppose we have 100 records coming from the source. Now for a target column population we used a Sequence generator.

Suppose the Current Value is 0 and End Value of Sequence generator is set to 80. What will happen?

**Ans.** **End Value** is the maximum value the Sequence Generator will generate. After it reaches the End value the session fails with the following error message:

TT\_11009 Sequence Generator Transformation: Overflow error.

Failing of session can be handled if the Sequence Generator is configured to **Cycle** through the sequence, i.e. whenever the Integration Service reaches the configured end value for the sequence, it wraps around and starts the cycle again, beginning with the configured Start Value.

**Q35.** What are the changes we observe when we promote a non resuable Sequence Generator to a resuable one? And what happens if we set the Number of Cached Values to 0 for a reusable transformation?

**Ans.** When we convert a non reusable sequence generator to resuable one we observe that the **Number of Cached Values** is set to 1000 by default; And the **Reset** property is disabled.

When we try to set the **Number of Cached Values** property of a Reusable Sequence Generator to 0 in the Transformation Developer we encounter the following error message:

**The number of cached values must be greater than zero for reusable sequence transformation.**

### Revisiting Aggregator Transformation

**Q36.** What is an Aggregator Transformation?

**Ans.** An aggregator is an Active, Connected transformation which performs aggregate calculations like**AVG**, **COUNT**, **FIRST**, **LAST**, **MAX**, **MEDIAN**, **MIN**, **PERCENTILE**, **STDDEV**, **SUM** and**VARIANCE**.

**Q37.** How an Expression Transformation differs from Aggregator Transformation?

**Ans.** An Expression Transformation performs calculation on a **row-by-row** basis. An Aggregator Transformation performs calculations **on groups**.

**Q38.** Does an Informatica Transformation support only Aggregate expressions?

**Ans.** Apart from aggregate expressions Informatica Aggregator also supports non-aggregate expressions and conditional clauses.

**Q39.** How does Aggregator Transformation handle NULL values?

**Ans.** By default, the aggregator transformation treats null values as NULL in aggregate functions. But we can specify to treat null values in aggregate functions as NULL or zero.

**Q40.** What is Incremental Aggregation?

**Ans.** We can enable the session option, Incremental Aggregation for a session that includes an Aggregator Transformation. When the Integration Service performs incremental aggregation, it actually passes changed source data through the mapping and uses the historical cache data to perform aggregate calculations incrementally.

For reference check [Implementing Informatica Incremental Aggregation](https://dwbi.org/etl/informatica/144-implementing-informaticas-incremental-aggregation)

**Q41.** What are the performance considerations when working with Aggregator Transformation?

**Ans.**

* Filter the unnecessary data before aggregating it. Place a Filter transformation in the mapping before the Aggregator transformation to reduce unnecessary aggregation.
* Improve performance by connecting only the necessary input/output ports to subsequent transformations, thereby reducing the size of the data cache.
* Use Sorted input which reduces the amount of data cached and improves session performance.

**Q42.** What differs when we choose Sorted Input for Aggregator Transformation?

**Ans.** Integration Service creates the index and data caches files in memory to process the Aggregator transformation. If the Integration Service requires more space as allocated for the index and data cache sizes in the transformation properties, it stores overflow values in cache files i.e. paging to disk. One way to increase session performance is to increase the index and data cache sizes in the transformation properties. But when we check Sorted Input the Integration Service uses memory to process an Aggregator transformation it does not use cache files.

**Q43.** Under what conditions selecting Sorted Input in aggregator will still not boost session performance?

**Ans.**

* Incremental Aggregation, session option is enabled.
* The aggregate expression contains nested aggregate functions.
* Source data is data driven.

**Q44.** Under what condition selecting Sorted Input in aggregator may fail the session?

**Ans.**

* If the input data is not sorted correctly, the session will fail.
* Also if the input data is properly sorted, the session may fail if the sort order by ports and the group by ports of the aggregator are not in the same order.

**Q45.** Suppose we do not group by on any ports of the aggregator what will be the output.

**Ans.** If we do not group values, the Integration Service will return **only the last row** for the input rows.

**Q46.** What is the expected value if the column in an aggregator transform is neither a group by nor an aggregate expression?

**Ans.** Integration Service produces one row for each group based on the group by ports. The columns which are neither part of the key nor aggregate expression will return the corresponding value of last record of the group received. However, if we specify particularly the FIRST function, the Integration Service then returns the value of the specified first row of the group. So default is the **LAST** function.

**Q47.** Give one example for each of Conditional Aggregation, Non-Aggregate expression and Nested Aggregation.

**Ans.**

Use conditional clauses in the aggregate expression to reduce the number of rows used in the aggregation. The conditional clause can be any clause that evaluates to TRUE or FALSE.

SUM( SALARY, JOB = CLERK )

Use non-aggregate expressions in group by ports to modify or replace groups.

IIF( PRODUCT = Brown Bread, Bread, PRODUCT )

The expression can also include one aggregate function within another aggregate function, such as:

MAX( COUNT( PRODUCT ))

### Revisiting Rank Transformation

**Q48.** What is a Rank Transform?

**Ans.** Rank is an Active Connected Informatica transformation used to select a set of top or bottom values of data.

**Q49.** How does a Rank Transform differ from Aggregator Transform functions MAX and MIN?

**Ans.** Like the Aggregator transformation, the Rank transformation lets us group information. The Rank Transform allows us to select a group of top or bottom **values**, not just **one value** as in case of Aggregator MAX, MIN functions.

**Q50.** What is a RANK port and RANKINDEX?

**Ans.** Rank port is an input/output port use to specify the column for which we want to rank the source values. By default Informatica creates an output port RANKINDEX for each Rank transformation. It stores the ranking position for each row in a group.

**Q51.** How can you get ranks based on different groups?

**Ans.** Rank transformation lets us group information. We can configure one of its input/output ports as a group by port. For each unique value in the group port, the transformation creates a group of rows falling within the rank definition (top or bottom, and a particular number in each rank).

**Q52.** What happens if two rank values match?

**Ans.** If two rank values match, they receive the same value in the rank index and the transformation skips the next value.

**Q53.** What are the restrictions of Rank Transformation?

**Ans.**

* We can connect ports from only one transformation to the Rank transformation.
* We can select the top or bottom rank.
* We need to select the Number of records in each rank.
* We can designate only one Rank port in a Rank transformation.

**Q54.** How does a Rank Cache works?

**Ans.** During a session, the Integration Service compares an input row with rows in the data cache. If the input row out-ranks a cached row, the Integration Service replaces the cached row with the input row. If we configure the Rank transformation to rank based on different groups, the Integration Service ranks incrementally for each group it finds. The Integration Service creates an index cache to stores the group information and data cache for the row data.

**Q55.** How does Rank transformation handle string values?

**Ans.** Rank transformation can return the strings at the top or the bottom of a session sort order. When the Integration Service runs in Unicode mode, it sorts character data in the session using the selected sort order associated with the Code Page of IS which may be French, German, etc. When the Integration Service runs in ASCII mode, it ignores this setting and uses a binary sort order to sort character data.

### Revisiting Sorter Transformation

**Q56.** What is a Sorter Transformation?

**Ans.** Sorter Transformation is an Active, Connected Informatica transformation used to sort data in ascending or descending order according to specified sort keys. The Sorter transformation contains only input/output ports.

**Q57.** Why is Sorter an Active Transformation?

**Ans.** When the Sorter transformation is configured to treat output rows as distinct, it assigns all ports as part of the sort key. The Integration Service discards duplicate rows compared during the sort operation. The number of Input Rows will vary as compared with the Output rows and hence it is an Active transformation.

**Q58.** How does Sorter handle Case Sensitive sorting?

**Ans.** The Case Sensitive property determines whether the Integration Service considers case when sorting data. When we enable the Case Sensitive property, the Integration Service sorts uppercase characters higher than lowercase characters.

**Q59.** How does Sorter handle NULL values?

**Ans.** We can configure the way the Sorter transformation treats null values. Enable the property Null Treated Low if we want to treat null values as lower than any other value when it performs the sort operation. Disable this option if we want the Integration Service to treat null values as higher than any other value.

**Q60.** How does a Sorter Cache works?

**Ans.** The Integration Service passes all incoming data into the Sorter Cache before Sorter transformation performs the sort operation.

The Integration Service uses the Sorter Cache Size property to determine the maximum amount of memory it can allocate to perform the sort operation. If it cannot allocate enough memory, the Integration Service fails the session. For best performance, configure Sorter cache size with a value less than or equal to the amount of available physical RAM on the Integration Service machine.

If the amount of incoming data is greater than the amount of Sorter cache size, the Integration Service temporarily stores data in the Sorter transformation work directory. The Integration Service requires disk space of at least twice the amount of incoming data when storing data in the work directory.

### Revisiting Union Transformation

**Q61.** What is a Union Transformation?

**Ans.** The Union transformation is an Active, Connected non-blocking multiple input group transformation use to merge data from multiple pipelines or sources into one pipeline branch. Similar to the UNION ALL SQL statement, the Union transformation does not remove duplicate rows.

**Q62.** What are the restrictions of Union Transformation?

**Ans.**

* All input groups and the output group must have matching ports. The precision, datatype, and scale must be identical across all groups.
* We can create multiple input groups, but only one default output group.
* The Union transformation does not remove duplicate rows.
* We cannot use a Sequence Generator or Update Strategy transformation upstream from a Union transformation.
* The Union transformation does not generate transactions.

### General questions

**Q63.** What is the difference between Static and Dynamic Lookup Cache?

**Ans.** We can configure a Lookup transformation to cache the corresponding lookup table. In case of static or read-only lookup cache the Integration Service caches the lookup table at the beginning of the session and does not update the lookup cache while it processes the Lookup transformation.

In case of dynamic lookup cache the Integration Service dynamically inserts or updates data in the lookup cache and passes the data to the target. The dynamic cache is synchronized with the target.

**Q64.** What is Persistent Lookup Cache?

**Ans.** Lookups are cached by default in Informatica. Lookup cache can be either non-persistent or persistent. The Integration Service saves or deletes lookup cache files after a successful session run based on whether the Lookup cache is checked as persistent or not.

**Q65.** What is the difference between Reusable transformation and Mapplet?

**Ans.** Any Informatica Transformation created in the in the Transformation Developer or a non-reusable promoted to reusable transformation from the mapping designer which can be used in multiple mappings is known as Reusable Transformation. When we add a reusable transformation to a mapping, we actually add an instance of the transformation. Since the instance of a reusable transformation is a pointer to that transformation, when we change the transformation in the Transformation Developer, its instances reflect these changes.

A Mapplet is a reusable object created in the Mapplet Designer which contains a **set of transformations**and lets us reuse the transformation logic in multiple mappings. A Mapplet can contain as many transformations as we need. Like a reusable transformation when we use a mapplet in a mapping, we use an instance of the mapplet and any change made to the mapplet is inherited by all instances of the mapplet.

**Q66.** What are the transformations that are not supported in Mapplet?

**Ans.** Normalizer, Cobol sources, XML sources, XML Source Qualifier transformations, Target definitions, Pre- and post- session Stored Procedures, Other Mapplets.

**Q67.** What are the ERROR tables present in Informatica?

**Ans.**

* **PMERR\_DATA**- Stores data and metadata about a transformation row error and its corresponding source row.
* **PMERR\_MSG**- Stores metadata about an error and the error message.
* **PMERR\_SESS**- Stores metadata about the session.
* **PMERR\_TRANS**- Stores metadata about the source and transformation ports, such as name and datatype, when a transformation error occurs.

**Q68.** What is the difference between STOP and ABORT?

**Ans.** When we issue the STOP command on the executing session task, the Integration Service stops reading data from source. It continues processing, writing and committing the data to targets. If the Integration Service cannot finish processing and committing data, we can issue the abort command.

In contrast ABORT command has a timeout period of 60 seconds. If the Integration Service cannot finish processing and committing data within the timeout period, it kills the DTM process and terminates the session.

**Q69.** Can we copy a session to new folder or new repository?

**Ans.** Yes we can copy session to new folder or repository provided the corresponding Mapping is already in there.

**Q70.** What type of join does Lookup support?

**Ans.** Lookup is just similar like SQL LEFT OUTER JOIN.

***Data Warehouse:***

## What is the difference between OLTP and OLAP?

OLTP is the transaction system that collects business data. Whereas OLAP is the reporting and analysis system on that data.

OLTP systems are optimized for INSERT, UPDATE operations and therefore highly normalized. On the other hand, OLAP systems are deliberately denormalized for fast data retrieval through SELECT operations.

**Explanatory Note:**

In a departmental shop, when we pay the prices at the check-out counter, the sales person at the counter keys-in all the data into a "Point-Of-Sales" machine. That data is transaction data and the related system is a OLTP system.   
  
On the other hand, the manager of the store might want to view a report on out-of-stock materials, so that he can place purchase order for them. Such report will come out from OLAP system.

## What is data mart?

Data marts are generally designed for a single subject area. An organization may have data pertaining to different departments like Finance, HR, Marketing etc. stored in data warehouse and each department may have separate data marts. These data marts can be built on top of the data warehouse.

## What is ER model?

ER model or entity-relationship model is a particular methodology of data modeling wherein the goal of modeling is to normalize the data by reducing redundancy. This is different than dimensional modeling where the main goal is to improve the data retrieval mechanism.

## What is dimensional modeling?

Dimensional model consists of dimension and fact tables. Fact tables store different transactional measurements and the foreign keys from dimension tables that qualifies the data. The goal of Dimensional model is **not** to achieve high degree of normalization but to facilitate easy and faster data retrieval.

Ralph Kimball is one of the strongest proponents of this very popular data modeling technique which is often used in many enterprise level data warehouses.

If you want to read a quick and simple guide on dimensional modeling, please check our [Guide to dimensional modeling](https://dwbi.org/data-modelling/dimensional-model/1-dimensional-modeling-guide.html).

## What is dimension?

A dimension is something that qualifies a quantity (measure).

For an example, consider this: If I just say… “20kg”, it does not mean anything. But if I say, "20kg of Rice (Product) is sold to Ramesh (customer) on 5th April (date)", then that gives a meaningful sense. These*product, customer* and *dates* are some dimension that qualified the measure - 20kg.

Dimensions are mutually independent. Technically speaking, a dimension is a data element that categorizes each item in a data set into non-overlapping regions.

## What is Fact?

A fact is something that is quantifiable (Or measurable). Facts are typically (but not always) numerical values that can be aggregated.

## What are additive, semi-additive and non-additive measures?

#### Non-additive Measures

Non-additive measures are those which can not be used inside any numeric aggregation function (e.g. SUM(), AVG() etc.). One example of non-additive fact is any kind of ratio or percentage. Example, 5% profit margin, revenue to asset ratio etc. A non-numerical data can also be a non-additive measure when that data is stored in fact tables, e.g. some kind of varchar flags in the fact table.

#### Semi Additive Measures

Semi-additive measures are those where only a subset of aggregation function can be applied. Let’s say account balance. A sum() function on balance does not give a useful result but max() or min() balance might be useful. Consider price rate or currency rate. Sum is meaningless on rate; however, average function might be useful.

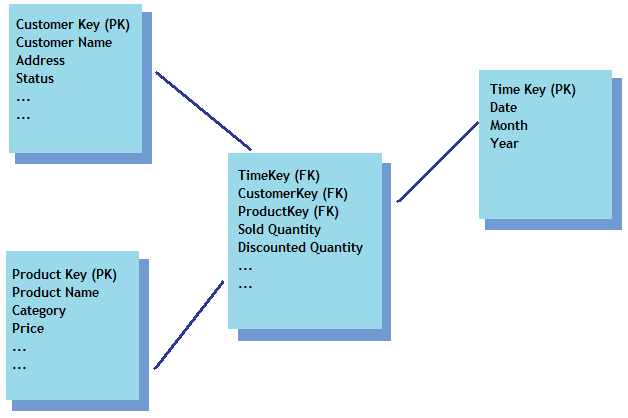
#### Additive Measures

Additive measures can be used with any aggregation function like Sum(), Avg() etc. Example is Sales Quantity etc.

At this point, I will request you to pause and make some time to read this article on ["Classifying data for successful modeling"](https://dwbi.org/data-modelling/dimensional-model/16-classifying-data-for-successful-modeling.html). This article helps you to understand the differences between dimensional data/ factual data etc. from a fundamental perspective

## What is Star-schema?

This schema is used in data warehouse models where one centralized fact table references number of dimension tables so as the keys (primary key) from all the dimension tables flow into the fact table (as foreign key) where measures are stored. This entity-relationship diagram looks like a star, hence the name.



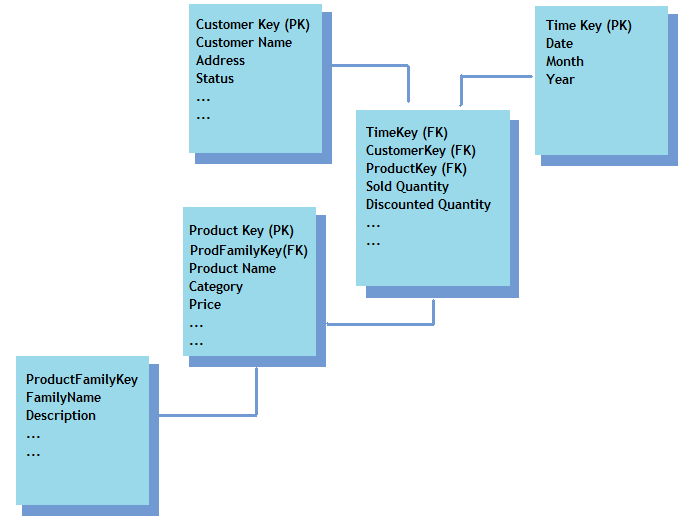
Consider a fact table that stores sales quantity for each product and customer on a certain time. Sales quantity will be the measure here and keys from customer, product and time dimension tables will flow into the fact table.

If you are not very familiar about Star Schema design or its use, we strongly recommend you read our excellent article on this subject - [different schema in dimensional modeling](https://dwbi.org/data-modelling/dimensional-model/17-dimensional-modeling-schema.html)

## What is snow-flake schema?

This is another logical arrangement of tables in dimensional modeling where a centralized fact table references number of other dimension tables; however, those dimension tables are further normalized into multiple related tables.

Consider a fact table that stores sales quantity for each product and customer on a certain time. Sales quantity will be the measure here and keys from customer, product and time dimension tables will flow into the fact table. Additionally all the products can be further grouped under different product families stored in a different table so that primary key of product family tables also goes into the product table as a foreign key. Such construct will be called a snow-flake schema as product table is further snow-flaked into product family.



**Note**  
Snow-flake increases degree of normalization in the design.

## What are the different types of dimension?

In a data warehouse model, dimension can be of following types,

1. Conformed Dimension
2. Junk Dimension
3. Degenerated Dimension
4. Role Playing Dimension

Based on how frequently the data inside a dimension changes, we can further classify dimension as

1. Unchanging or static dimension (UCD)
2. Slowly changing dimension (SCD)
3. Rapidly changing Dimension (RCD)

You may also read, [Modeling for various slowly changing dimension](https://dwbi.org/data-modelling/dimensional-model/19-modeling-for-various-slowly-changing-dimension.html) and [Implementing Rapidly changing dimension](https://dwbi.org/data-modelling/dimensional-model/20-implementing-rapidly-changing-dimension.html) to know more about SCD, RCD dimensions etc.

## What is a 'Conformed Dimension'?

A conformed dimension is the dimension that is shared across multiple subject area. Consider 'Customer' dimension. Both marketing and sales department may use the same customer dimension table in their reports. Similarly, a 'Time' or 'Date' dimension will be shared by different subject areas. These dimensions are conformed dimension.

Theoretically, two dimensions which are either identical or strict mathematical subsets of one another are said to be conformed.

## What is degenerated dimension?

A degenerated dimension is a dimension that is derived from fact table and does not have its own dimension table.

A dimension key, such as transaction number, receipt number, Invoice number etc. does not have any more associated attributes and hence can not be designed as a dimension table.

## What is junk dimension?

A junk dimension is a grouping of typically low-cardinality attributes (flags, indicators etc.) so that those can be removed from other tables and can be junked into an abstract dimension table.

These junk dimension attributes might not be related. The only purpose of this table is to store all the combinations of the dimensional attributes which you could not fit into the different dimension tables otherwise. Junk dimensions are often used to implement [Rapidly Changing Dimensions](https://dwbi.org/data-modelling/dimensional-model/20-implementing-rapidly-changing-dimension.html) in data warehouse.

## What is a role-playing dimension?

Dimensions are often reused for multiple applications within the same database with different contextual meaning. For instance, a "Date" dimension can be used for "Date of Sale", as well as "Date of Delivery", or "Date of Hire". This is often referred to as a 'role-playing dimension'

## What is SCD?

SCD stands for slowly changing dimension, i.e. the dimensions where data is slowly changing. These can be of many types, e.g. Type 0, Type 1, Type 2, Type 3 and Type 6, although Type 1, 2 and 3 are most common. Read [this](https://dwbi.org/data-modelling/dimensional-model/19-modeling-for-various-slowly-changing-dimension.html) article to gather in-depth knowledge on various SCD tables.

## What is rapidly changing dimension?

This is a dimension where data changes rapidly. Read [this](https://dwbi.org/data-modelling/dimensional-model/20-implementing-rapidly-changing-dimension.html) article to know how to implement RCD.

## Describe different types of slowly changing Dimension (SCD)

Type 0:

A Type 0 dimension is where dimensional changes are not considered. This does not mean that the attributes of the dimension do not change in actual business situation. It just means that, even if the value of the attributes change, history is not kept and the table holds all the previous data.

Type 1:

A type 1 dimension is where history is not maintained and the table always shows the recent data. This effectively means that such dimension table is always updated with recent data whenever there is a change, and because of this update, we lose the previous values.

Type 2:

A type 2 dimension table tracks the historical changes by creating separate rows in the table with different surrogate keys. Consider there is a customer C1 under group G1 first and later on the customer is changed to group G2. Then there will be two separate records in dimension table like below,

| **Key** | **Customer** | **Group** | **Start Date** | **End Date** |
| --- | --- | --- | --- | --- |
| 1 | C1 | G1 | 1st Jan 2000 | 31st Dec 2005 |
| 2 | C1 | G2 | 1st Jan 2006 | *NULL* |

Note that separate surrogate keys are generated for the two records. NULL end date in the second row denotes that the record is the current record. Also note that, instead of start and end dates, one could also keep version number column (1, 2 … etc.) to denote different versions of the record.

Type 3:

A type 3 dimension stored the history in a separate column instead of separate rows. So unlike a type 2 dimension which is vertically growing, a type 3 dimension is horizontally growing. See the example below,

| **Key** | **Customer** | **Previous Group** | **Current Group** |
| --- | --- | --- | --- |
| 1 | C1 | G1 | G2 |

This is only good when you need not store many consecutive histories and when date of change is not required to be stored.

Type 6:

A type 6 dimension is a hybrid of type 1, 2 and 3 (1+2+3) which acts very similar to type 2, but only you add one extra column to denote which record is the current record.

| **Key** | **Customer** | **Group** | **Start Date** | **End Date** | **Current Flag** |
| --- | --- | --- | --- | --- | --- |
| 1 | C1 | G1 | 1st Jan 2000 | 31st Dec 2005 | N |
| 2 | C1 | G2 | 1st Jan 2006 | *NULL* | Y |

## What is a mini dimension?

Mini dimensions can be used to handle rapidly changing dimension scenario. If a dimension has a huge number of rapidly changing attributes it is better to separate those attributes in different table called mini dimension. This is done because if the main dimension table is designed as SCD type 2, the table will soon outgrow in size and create performance issues. It is better to segregate the rapidly changing members in different table thereby keeping the main dimension table small and performing.

## What is a fact-less-fact?

A fact table that does not contain any measure is called a fact-less fact. This table will only contain keys from different dimension tables. This is often used to resolve a many-to-many cardinality issue.

**Explanatory Note:**

Consider a school, where a single student may be taught by many teachers and a single teacher may have many students. To model this situation in dimensional model, one might introduce a fact-less-fact table joining teacher and student keys. Such a fact table will then be able to answer queries like,

1. Who are the students taught by a specific teacher.
2. Which teacher teaches maximum students.
3. Which student has highest number of teachers.etc. etc.

## What is a coverage fact?

A fact-less-fact table can only answer 'optimistic' queries (positive query) but can not answer a negative query. Again consider the illustration in the above example. A fact-less fact containing the keys of tutors and students can not answer a query like below,

1. Which teacher did **not** teach any student?
2. Which student was **not** taught by any teacher?

Why not? Because fact-less fact table only stores the positive scenarios (like student being taught by a tutor) but if there is a student who is **not** being taught by a teacher, then that student's key does not appear in this table, thereby reducing the coverage of the table.

Coverage fact table attempts to answer this - often by adding an extra flag column. Flag = 0 indicates a negative condition and flag = 1 indicates a positive condition. To understand this better, let's consider a class where there are 100 students and 5 teachers. So coverage fact table will ideally store 100 X 5 = 500 records (all combinations) and if a certain teacher is not teaching a certain student, the corresponding flag for that record will be 0.

## What are incident and snapshot facts

A fact table stores some kind of measurements. Usually these measurements are stored (or captured) against a specific time and these measurements vary with respect to time. Now it might so happen that the business might not able to capture all of its measures always for every point in time. Then those unavailable measurements can be kept empty (Null) or can be filled up with the last available measurements. The first case is the example of incident fact and the second one is the example of snapshot fact.

## What is aggregation and what is the benefit of aggregation?

A data warehouse usually captures data with same degree of details as available in source. The "degree of detail" is termed as granularity. But all reporting requirements from that data warehouse do not need the same degree of details.

To understand this, let's consider an example from retail business. A certain retail chain has 500 shops accross Europe. All the shops record detail level transactions regarding the products they sale and those data are captured in a data warehouse.

Each shop manager can access the data warehouse and they can see which products are sold by whom and in what quantity on any given date. Thus the data warehouse helps the shop managers with the detail level data that can be used for inventory management, trend prediction etc.

Now think about the CEO of that retail chain. He does not really care about which certain sales girl in London sold the highest number of chopsticks or which shop is the best seller of 'brown breads'. All he is interested is, perhaps to check the percentage increase of his revenue margin across Europe. Or may be year to year sales growth on eastern Europe. Such data is aggregated in nature. Because Sales of goods in East Europe is derived by summing up the individual sales data from each shop in East Europe.

Therefore, to support different levels of data warehouse users, data aggregation is needed.

## What is slicing-dicing?

Slicing means showing the slice of a data, given a certain set of dimension (e.g. Product) and value (e.g. Brown Bread) and measures (e.g. sales).

Dicing means viewing the slice with respect to different dimensions and in different level of aggregations.

Slicing and dicing operations are part of pivoting.

## What is drill-through?

Drill through is the process of going to the detail level data from summary data.

Consider the above example on retail shops. If the CEO finds out that sales in East Europe has declined this year compared to last year, he then might want to know the root cause of the decrease. For this, he may start drilling through his report to more detail level and eventually find out that even though individual shop sales has actually increased, the overall sales figure has decreased because a certain shop in Turkey has stopped operating the business. The detail level of data, which CEO was not much interested on earlier, has this time helped him to pin point the root cause of declined sales. And the method he has followed to obtain the details from the aggregated data is called drill through.

## What is the difference between inner and outer join? Explain with example.

#### Inner Join

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).

Inner join returns rows when there is at least one match in both tables

If none of the record matches between two tables, then INNER JOIN will return a NULL set. Below is an example of INNER JOIN and the resulting set.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE

FROM DEPT dept, EMPLOYEE emp

WHERE emp.dept\_id = dept.id

| **Department** | **Employee** |
| --- | --- |
| HR | Inno |
| HR | Privy |
| Engineering | Robo |
| Engineering | Hash |
| Engineering | Anno |
| Engineering | Darl |
| Marketing | Pete |
| Marketing | Meme |
| Sales | Tomiti |
| Sales | Bhuti |

#### Outer Join

Outer Join, on the other hand, will return matching rows from both tables as well as any unmatched rows from one or both the tables (based on whether it is single outer or full outer join respectively).

Outer Join can be full outer or single outer

Notice in our record set that there is no employee in the department 5 (Logistics). Because of this if we perform inner join, then Department 5 does not appear in the above result. However in the below query we perform an outer join (dept left outer join emp), and we can see this department.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE

FROM DEPT dept, EMPLOYEE emp

WHERE dept.id = emp.dept\_id (+)

| **Department** | **Employee** |
| --- | --- |
| HR | Inno |
| HR | Privy |
| Engineering | Robo |
| Engineering | Hash |
| Engineering | Anno |
| Engineering | Darl |
| Marketing | Pete |
| Marketing | Meme |
| Sales | Tomiti |
| Sales | Bhuti |
| Logistics |  |

The (+) sign on the emp side of the predicate indicates that emp is the outer table here. The above SQL can be alternatively written as below (will yield the same result as above):

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE

FROM DEPT dept LEFT OUTER JOIN EMPLOYEE emp

ON dept.id = emp.dept\_id

## What is the difference between JOIN and UNION?

SQL JOIN allows us to “lookup” records on other table based on the given conditions between two tables. For example, if we have the department ID of each employee, then we can use this department ID of the employee table to join with the department ID of department table to lookup department names.

UNION operation allows us to add 2 similar data sets to create resulting data set that contains all the data from the source data sets. Union does not require any condition for joining. For example, if you have 2 employee tables with same structure, you can UNION them to create one result set that will contain all the employees from both of the tables.

SELECT \* FROM EMP1

UNION

SELECT \* FROM EMP2;

## What is the difference between UNION and UNION ALL?

UNION and UNION ALL both unify for add two structurally similar data sets, but UNION operation returns only the unique records from the resulting data set whereas UNION ALL will return all the rows, even if one or more rows are duplicated to each other.

In the following example, I am choosing exactly the same employee from the emp table and performing UNION and UNION ALL. Check the difference in the result.

SELECT \* FROM EMPLOYEE WHERE ID = 5

UNION ALL

SELECT \* FROM EMPLOYEE WHERE ID = 5

| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| --- | --- | --- | --- | --- | --- |
| 5.0 | 2.0 | 2.0 | Anno | 80.0 | 01-Feb-2012 |
| 5.0 | 2.0 | 2.0 | Anno | 80.0 | 01-Feb-2012 |

SELECT \* FROM EMPLOYEE WHERE ID = 5

UNION

SELECT \* FROM EMPLOYEE WHERE ID = 5

| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| --- | --- | --- | --- | --- | --- |
| 5.0 | 2.0 | 2.0 | Anno | 80.0 | 01-Feb-2012 |

## What is the difference between WHERE clause and HAVING clause?

WHERE and HAVING both filters out records based on one or more conditions. The difference is, WHERE clause can only be applied on a static non-aggregated column whereas we will need to use HAVING for aggregated columns.

To understand this, consider this example.   
Suppose we want to see only those departments where department ID is greater than 3. There is no aggregation operation and the condition needs to be applied on a static field. We will use WHERE clause here:

SELECT \* FROM DEPT WHERE ID > 3

| **ID** | **NAME** |
| --- | --- |
| 4 | Sales |
| 5 | Logistics |

Next, suppose we want to see only those Departments where Average salary is greater than 80. Here the condition is associated with a non-static aggregated information which is “average of salary”. We will need to use HAVING clause here:

SELECT dept.name DEPARTMENT, avg(emp.sal) AVG\_SAL

FROM DEPT dept, EMPLOYEE emp

WHERE dept.id = emp.dept\_id (+)

GROUP BY dept.name

HAVING AVG(emp.sal) > 80

| **DEPARTMENT** | **AVG\_SAL** |
| --- | --- |
| Engineering | 90 |

As you see above, there is only one department (Engineering) where average salary of employees is greater than 80.

## What is the difference among UNION, MINUS and INTERSECT?

UNION combines the results from 2 tables and eliminates duplicate records from the result set.

MINUS operator when used between 2 tables, gives us all the rows from the first table except the rows which are present in the second table.

INTERSECT operator returns us only the matching or common rows between 2 result sets.

To understand these operators, let’s see some examples. We will use two different queries to extract data from our emp table and then we will perform UNION, MINUS and INTERSECT operations on these two sets of data.

#### UNION

SELECT \* FROM EMPLOYEE WHERE ID = 5

UNION

SELECT \* FROM EMPLOYEE WHERE ID = 6

| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| --- | --- | --- | --- | --- | --- |
| 5 | 2 | 2.0 | Anno | 80.0 | 01-Feb-2012 |
| 6 | 2 | 2.0 | Darl | 80.0 | 11-Feb-2012 |

#### MINUS

SELECT \* FROM EMPLOYEE

MINUS

SELECT \* FROM EMPLOYEE WHERE ID > 2

| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| --- | --- | --- | --- | --- | --- |
| 1 |  | 2 | Hash | 100.0 | 01-Jan-2012 |
| 2 | 1 | 2 | Robo | 100.0 | 01-Jan-2012 |

#### INTERSECT

SELECT \* FROM EMPLOYEE WHERE ID IN (2, 3, 5)

INTERSECT

SELECT \* FROM EMPLOYEE WHERE ID IN (1, 2, 4, 5)

| **ID** | **MGR\_ID** | **DEPT\_ID** | **NAME** | **SAL** | **DOJ** |
| --- | --- | --- | --- | --- | --- |
| 5 | 2 | 2 | Anno | 80.0 | 01-Feb-2012 |
| 2 | 1 | 2 | Robo | 100.0 | 01-Jan-2012 |

## What is Self Join and why is it required?

Self Join is the act of joining one table with itself.

Self Join is often very useful to convert a hierarchical structure into a flat structure

In our employee table example above, we have kept the manager ID of each employee in the same row as that of the employee. This is an example of how a hierarchy (in this case employee-manager hierarchy) is stored in the RDBMS table. Now, suppose if we need to print out the names of the manager of each employee right beside the employee, we can use self join. See the example below:

SELECT e.name EMPLOYEE, m.name MANAGER

FROM EMPLOYEE e, EMPLOYEE m

WHERE e.mgr\_id = m.id (+)

| **EMPLOYEE** | **MANAGER** |
| --- | --- |
| Pete | Hash |
| Darl | Hash |
| Inno | Hash |
| Robo | Hash |
| Tomiti | Robo |
| Anno | Robo |
| Privy | Robo |
| Meme | Pete |
| Bhuti | Tomiti |
| Hash |  |

The only reason we have performed a left outer join here (instead of INNER JOIN) is we have one employee in this table without a manager (employee ID = 1). If we perform inner join, this employee will not show-up.

## How can we transpose a table using SQL (changing rows to column or vice-versa) ?

The usual way to do it in SQL is to use CASE statement or DECODE statement.

## How to generate row number in SQL Without ROWNUM

Generating a row number – that is a running sequence of numbers for each row is not easy using plain SQL. In fact, the method I am going to show below is not very generic either. This method only works if there is at least one unique column in the table. This method will also work if there is no single unique column, but collection of columns that is unique. Anyway, here is the query:

SELECT name, sal, (SELECT COUNT(\*) FROM EMPLOYEE i WHERE o.name >= i.name) row\_num

FROM EMPLOYEE o

order by row\_num

| **NAME** | **SAL** | **ROW\_NUM** |
| --- | --- | --- |
| Anno | 80 | 1 |
| Bhuti | 60 | 2 |
| Darl | 80 | 3 |
| Hash | 100 | 4 |
| Inno | 50 | 5 |
| Meme | 60 | 6 |
| Pete | 70 | 7 |
| Privy | 50 | 8 |
| Robo | 100 | 9 |
| Tomiti | 70 | 10 |

The column that is used in the row number generation logic is called “sort key”. Here sort key is “name” column. For this technique to work, the sort key needs to be unique. We have chosen the column “name” because this column happened to be unique in our Employee table. If it was not unique but some other collection of columns was, then we could have used those columns as our sort key (by concatenating those columns to form a single sort key).

Also notice how the rows are sorted in the result set. We have done an explicit sorting on the row\_num column, which gives us all the row numbers in the sorted order. But notice that name column is also sorted (which is probably the reason why this column is referred as sort-key). If you want to change the order of the sorting from ascending to descending, you will need to change “>=” sign to “<=” in the query.

As I said before, this method is not very generic. This is why many databases already implement other methods to achieve this. For example, in Oracle database, every SQL result set contains a hidden column called ROWNUM. We can just explicitly select ROWNUM to get sequence numbers.

## How to select first 5 records from a table?

This question, often asked in many interviews, does not make any sense to me. The problem here is how do you define which record is first and which is second. Which record is retrieved first from the database is not deterministic. It depends on many uncontrollable factors such as how database works at that moment of execution etc. So the question should really be – “how to select any 5 records from the table?” But whatever it is, here is the solution:

In Oracle,

SELECT \*

FROM EMP

WHERE ROWNUM <= 5;

In SQL Server,

SELECT TOP 5 \* FROM EMP;

**Generic solution**,

I believe a generic solution can be devised for this problem if and only if there exists at least one distinct column in the table. For example, in our EMP table ID is distinct. We can use that distinct column in the below way to come up with a generic solution of this question that does not require database specific functions such as ROWNUM, TOP etc.

SELECT name

FROM EMPLOYEE o

WHERE (SELECT count(\*) FROM EMPLOYEE i WHERE i.name < o.name) < 5

| **name** |
| --- |
| Inno |
| Anno |
| Darl |
| Meme |
| Bhuti |

I have taken “name” column in the above example since “name” is happened to be unique in this table. I could very well take ID column as well.

In this example, if the chosen column was not distinct, we would have got more than 5 records returned in our output.

Do you have a better solution to this problem? If yes, post your solution in the comment.

## What is the difference between ROWNUM pseudo column and ROW\_NUMBER() function?

ROWNUM is a pseudo column present in Oracle database returned result set prior to ORDER BY being evaluated. So ORDER BY ROWNUM does not work.

ROW\_NUMBER() is an analytical function which is used in conjunction to OVER() clause wherein we can specify ORDER BY and also PARTITION BY columns.

Suppose if you want to generate the row numbers in the order of ascending employee salaries for example, ROWNUM will not work. But you may use ROW\_NUMBER() OVER() like shown below:

SELECT name, sal, row\_number() over(order by sal desc) rownum\_by\_sal

FROM EMPLOYEE o

| **name** | **Sal** | **ROWNUM\_BY\_SAL** |
| --- | --- | --- |
| Hash | 100 | 1 |
| Robo | 100 | 2 |
| Anno | 80 | 3 |
| Darl | 80 | 4 |
| Tomiti | 70 | 5 |
| Pete | 70 | 6 |
| Bhuti | 60 | 7 |
| Meme | 60 | 8 |
| Inno | 50 | 9 |
| Privy | 50 | 10 |

### What are the differences among ROWNUM, RANK and DENSE\_RANK?

ROW\_NUMBER assigns contiguous, unique numbers from 1.. N to a result set.

RANK does not assign unique numbers—nor does it assign contiguous numbers. If two records tie for second place, no record will be assigned the 3rd rank as no one came in third, according to RANK. See below:

SELECT name, sal, rank() over(order by sal desc) rank\_by\_sal

FROM EMPLOYEE o

| **name** | **Sal** | **RANK\_BY\_SAL** |
| --- | --- | --- |
| Hash | 100 | 1 |
| Robo | 100 | 1 |
| Anno | 80 | 3 |
| Darl | 80 | 3 |
| Tomiti | 70 | 5 |
| Pete | 70 | 5 |
| Bhuti | 60 | 7 |
| Meme | 60 | 7 |
| Inno | 50 | 9 |
| Privy | 50 | 9 |

DENSE\_RANK, like RANK, does not assign unique numbers, but it does assign contiguous numbers. Even though two records tied for second place, there is a third-place record. See below:

SELECT name, sal, dense\_rank() over(order by sal desc) dense\_rank\_by\_sal

FROM EMPLOYEE o

| **name** | **Sal** | **DENSE\_RANK\_BY\_SAL** |
| --- | --- | --- |
| Hash | 100 | 1 |
| Robo | 100 | 1 |
| Anno | 80 | 2 |
| Darl | 80 | 2 |
| Tomiti | 70 | 3 |
| Pete | 70 | 3 |
| Bhuti | 60 | 4 |
| Meme | 60 | 4 |
| Inno | 50 | 5 |
| Privy | 50 | 5 |