**Difference between ETL and ELT?**

* ETL means first we extract a Data then we transform a data and eventually Load a data
* ELT means first we extract a data then load it into staging area and then transform a data but with SQL Expression tasks not with transformations task in you fact and dimension tables.
* ELT is better as DB engine can perform transformations more better then SSI packages.
* So in short, when using ETL the transformations are processed by the ETL tools, while in ELT the transformations are processed by the target datasource.

**What is MDM? How to implement?**

MDM is unified data model (Master data Management).Instead of having varied snapshot of important information it is advisable to have Master data for that entity and refer to the same

1. Identify Master data entity with below points

- Data needs to be less volatile compare to transaction data (Which changes less compare to facts)

- Cardinality is more (Number of records are more for i.e. customers)

- Data changes over time then it should be qualified for MDM

- Reuse of Data from MDM (Reusability is high)

**Ralph vs Inmon**

Kimball uses the dimensional model such as star schemas or snowflakes to organize the data in dimensional data warehouse while Inmon uses ER model in enterprise data warehouse. Inmon only uses dimensional model for data marts only while Kimball uses it for all data

Inmon uses data marts as physical separation from enterprise data warehouse and they are built for departmental uses. While in Kimball’s architecture, it is unnecessary to separate the data marts from the dimensional data warehouse.

In dimensional data warehouse of Kimball, analytic systems can access data directly. While in Inmon’ s architecture, analytic systems can only access data in enterprise data warehouse via data marts.

**Steps to design DW from OLTP**

In general, building any data warehouse consists of the following steps:

**Flow of Data:**

* Landing(Extraction from Source)
* Staging(Data sits in one format from source):Data staging is an area where you hold the data temporary on data warehouse server. Data staging includes following steps
* Source data extraction and data transformation ( restructuring )
* Data transformation (data cleansing, value transformation )

**Extracting** the transactional data from the data sources into a staging area

* Transforming the transactional data
* Loading the transformed data into a dimensional database
* Building pre-calculated summary values to speed up report generation
* Building (or purchasing) a front-end reporting tool

**Analysis**:

* Requirement gathering for BI reports and business models.
* Understanding the objective for which we are designing DW.

**Design**:

* Identify the source databases or data, understanding and defining hierarchies.
* Identifying measures, identifying dimensions and their hierarchies, designing and creating physical models by defining schema of fact and dimension tables.
* Design Staging area for Data.
* Granularity level of dataware house - Requirement speccifci
* CDC of given data.

**Data presentation**

* Area where data will be in normalized form.

**Data Gathering**:

* This is one of the important step in designing the good Data Warehouse system.This is step is nothing but ETL development i.e., Extraction Transformation and Loading. Here the Data is Extracted and by using the transformations we convert this data to normalized form.The normalized format is not so good for understanding purpose.So we go for Data Presentation Area. Data gathering phase involves extracting and cleaning data to prepare it for loading into the warehouse. This phase is called low level design.

**Data mart creation**:

* Here we convert the Normalized model to Dimensional model where the user can understand very easily. Here the data will be in data marts. By combining the data marts we can have a Data Warehouse.

Furthermore,

The last phase, **loading phase**, is to load the prepared data into the data warehouse.

* **Conformed Dimension**: In data warehousing, a conformed dimension is a dimension that has the same meaning to every fact with which it relates and which can be connected to multiple DW and give the same meaning.
* **Conformed Fact**: A conformed fact is a shared fact that is designed to be used in the same way. across multiple data marts.

**SCD Type:**

* **Type 1:** overwrite
* **Type 2**: add new row (With Is Active or Effective Dates Column)
* **Type 3**: add new attribute (Add new column to capture the change i.e having previous stateof customer and current state of customer)
* **Type 4**: add history table

**Benefits of Surrogate Keys:**

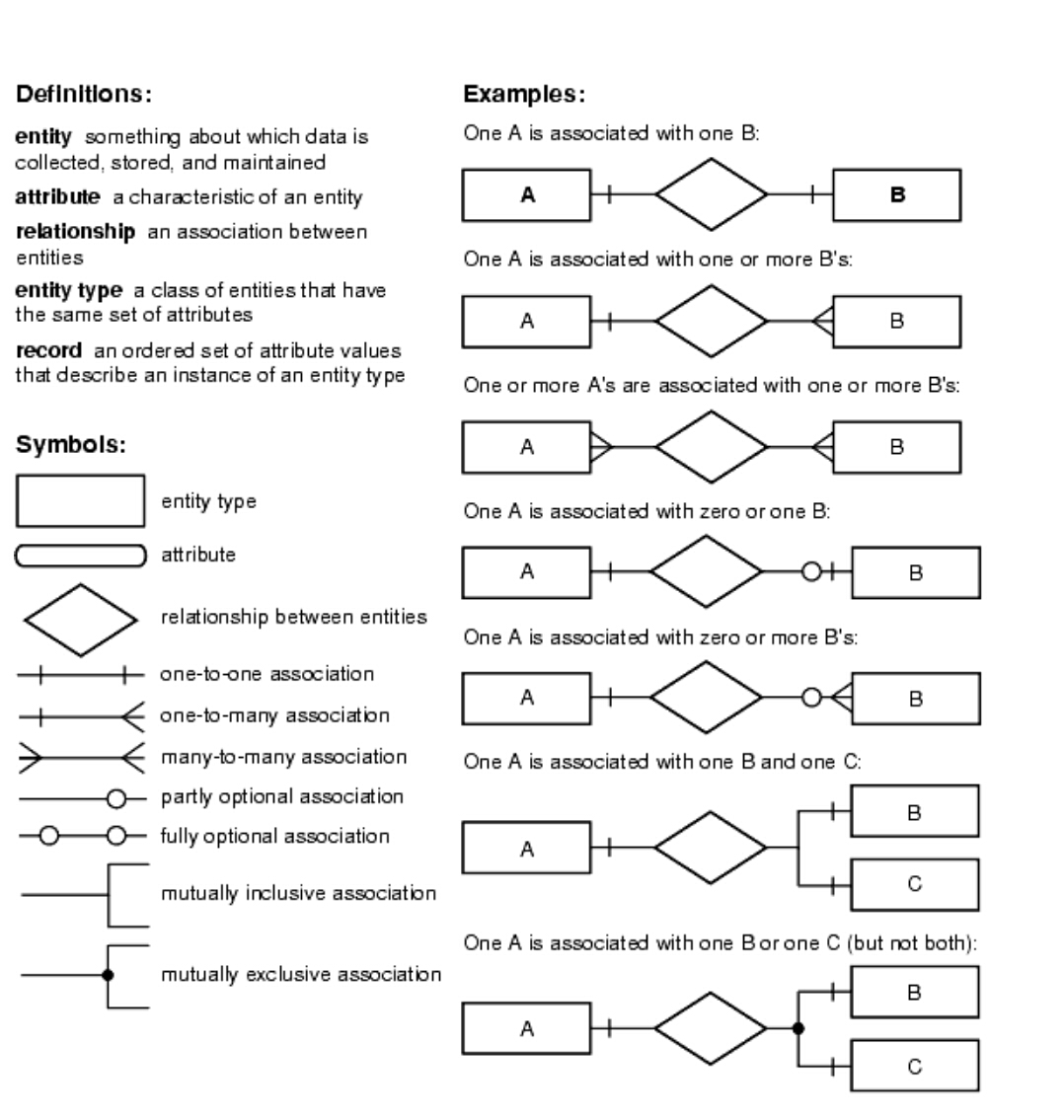
* Performance: Surrogate key is always numeric key unlike any production key which is alphanumeric so it can be good qualifier to create index and use in join statements
* No Possibilities of duplication and maintenance is quite easy as no composite key creation required.
* Immutability: Keys can’t be changed
* Stay as it is with requirement changes( Natural keys can become duplicate on merger surrogate cant)
* Surrogate key assignments
  + ODS (Transformations)
  + DW(Data presentation area)

**Fact less Fact**: Facts which does not contain any measures and Count keyword only we can operate.This table only contains IDs which links to dimension.

**Additive Facts:**

**Non-Additive facts:**

**Semi-Additive facts:**

**ER Diagram points:**

Informatica:

**Informatica Concepts to look:**

-error handling

-Identifying bottlenecks

-Source and Destination

-Nested join and Hash Joins

-Multiple sources in SSIS jobs

-How to optimize job which take huge amount of time?

-Challenging job

-Incremental load

-Protection level

-Deployment

**Assignments of Informatica:**

1. ~~Load a flat file to Database~~
2. ~~Load a database table to Flat file – Done~~
3. ~~Load a database table to Multiple targets - Done~~
4. Avoid double loads when jobs are run twice.
5. ~~Add calculated column with functions and expressions and load into Target - Done~~
6. Handle erroneous data into other table and clean into target table.
7. Join data from various sources through transformation and load into single target
8. Incremental Load of Data from Flat file
9. Change data capture implementation
10. Target Destructive loads
11. Update Existing Record or if not exists then insert new.
12. Remove Duplicate records from FF apart from using SORTER transformation.
13. Create query in source qualifier and create job
14. Filter Transformation
15. Router Transformation
16. Differences between Filter and Router
17. Sequence Generator Transformation
18. Cache
19. Sorter Transformation
20. Aggregator Transformation
21. Implement SCD Job Type 2
22. Implement MDM Job
23. Multigroup Transformations
24. Normalizer Transformation
25. Pivot and unpivot data
26. Compare previous value in expression transformation
27. Geenrate running total

**Transformations in Informatica:**

| **Transformation** | **Type** | **Description** |
| --- | --- | --- |
| [Aggregator](http://www.howtointegratedata.com/aggregator-transformation-in-informatica/) | Active | Performs aggregate calculations. |
| Application Source Qualifier | Active | Represents the rows that the Integration Service reads from an application, such as an ERP source, when it runs a session. |
| Custom | Active or Passive | Calls a procedure in a shared library or DLL. |
| Data Masking | Passive | Replaces sensitive production data with realistic test data for non-production environments. |
| [Expression](http://www.howtointegratedata.com/expression-transformation-in-informatica/) | Passive | Calculates a value. |
| External Procedure | Passive | Calls a procedure in a shared library or in the COM layer of Windows. |
| [Filter](http://www.howtointegratedata.com/filter-transformation-in-informatica/) | Active | Filters data. |
| HTTP | Passive | Connects to an HTTP server to read or update data. |
| Input | Passive | Defines mapplet input rows. Available in the Mapplet Designer. |
| [Joiner](http://www.howtointegratedata.com/joiner-transformation-informatica/) | Active | Joins data from different databases or flat file systems. |
| Lookup | Active or Passive | Look up and return data from a flat file, relational table, view, or synonym. |
| Normalizer | Active | Source qualifier for COBOL sources. Can also use in the pipeline to normalize data from relational or flat file sources. |
| Output | Passive | Defines mapplet output rows. Available in the Mapplet Designer. |
| [Rank](http://www.howtointegratedata.com/rank-transformation-in-informatica/) | Active | Limits records to a top or bottom range. |
| [Router](http://www.howtointegratedata.com/router-transformation-in-informatica/) | Active | Routes data into multiple transformations based on group conditions. |
| [Sequence Generator](http://www.howtointegratedata.com/sequence-generator-transformation-in-informatica/) | Passive | Generates primary keys. |
| [Sorter](http://www.howtointegratedata.com/sorter-transformation-in-informatica/) | Active | Sorts data based on a sort key. |
| [Source Qualifier](http://www.howtointegratedata.com/source-qualifier-transformation-in-informatica/) | Active | Represents the rows that the Integration Service reads from a relational or flat file source when it runs a session. |
| SQL | Active or Passive | Executes SQL queries against a database. |
| Stored Procedure | Passive | Calls a stored procedure. |
| Transaction Control | Active | Defines commit and rollback transactions.  Transaction Control Transformation is used in a mapping to control transactions according to the user-defined conditions.  Use expression transformation to form the condition.  Built-in variables in the Expression Editor when we create a transaction control expression:    **TC\_CONTINUE\_TRANSACTION**. The PowerCenter Server does not perform any transaction change for this row. This is the default value of the expression.  **TC\_COMMIT\_BEFORE**. The PowerCenter Server commits the transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.  **TC\_COMMIT\_AFTER**. The PowerCenter Server writes the current row to the target, commits the transaction, and begins a new transaction. The current row is in the committed transaction.  **TC\_ROLLBACK\_BEFORE**. The PowerCenter Server rolls back the current transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.  **TC\_ROLLBACK\_AFTER**. The PowerCenter Server writes the current row to the target, rolls back the transaction, and begins a new transaction. The current row is in the rolled back transaction.  Used for:   * Generating dynamic flat files on output |
| [Union](http://www.howtointegratedata.com/informatica-union-transformation/) | Active | Merges data from different databases or flat file systems. |
| Unstructured Data | Active or Passive | Transforms data in unstructured and semi-structured formats. |
| [Update Strategy](http://www.howtointegratedata.com/update-strategy-transformation-in-informatica/) | Active | Determines whether to insert, delete, update, or reject rows. |
| XML Generator | Active | Reads data from one or more input ports and outputs XML through a single output port. |
| XML Parser | Active | Reads XML from one input port and outputs data to one or more output ports. |
| XML Source Qualifier | Active | Represents the rows that the Integration Service reads from an XML source when it runs a session. |

**How to remove duplicate data from flat file informatica?**

* Use Sorter Transformation and distinct property
* If RDBMS use Source Qualifier distinct property
* Use aggregator with all column as keys
* Expression transformation and variable to mark duplicates and fitler to remove it

**Difference between Joiner and Look up?**

* In lookup we can override the query but in joiner we cannot.
* In lookup we can provide different types of operators like – “>,<,>=,<=,!=” but in joiner only “= “ (equal to )operator is available.
* In lookup we can restrict the number of rows while reading the relational table using lookup override but in joiner we cannot restrict the number of rows while reading.
* In joiner we can join the tables based on- Normal Join, Master Outer, Detail Outer and Full Outer Join but in lookup this facility is not available .Lookup behaves like Left Outer Join of database.

**How to create output file dynamically at job run time?**

* Use transaction control transformation and previous record check and select commit option and target as flat file which will generate file for each value.

**Union Transformation:**

* Informatica Union is always union all transformation
* Union transformation should have a similar source.
* Input and output source should be same.
* Use filter with union all transformation for unique records
* Works with multiple sources

**Normalizer Transformation:**

* Source qualifier for COBOL sources
* Generate single record multiple times.
* Convert Rows to Column (Pivoting)
* GKID and GCID:
  + GKID: generate sequence number each time processes the source row.
  + GCID: Generated column number- The GCID is an index for the instance of the multiple-occurring data.

**Store Procedure Transformation:**

Debugging is done easily in informatica compare to SP

**XML Transformation:**

XML Parser

XML Generator

XML Source Qualifier

**SQL Transformation**

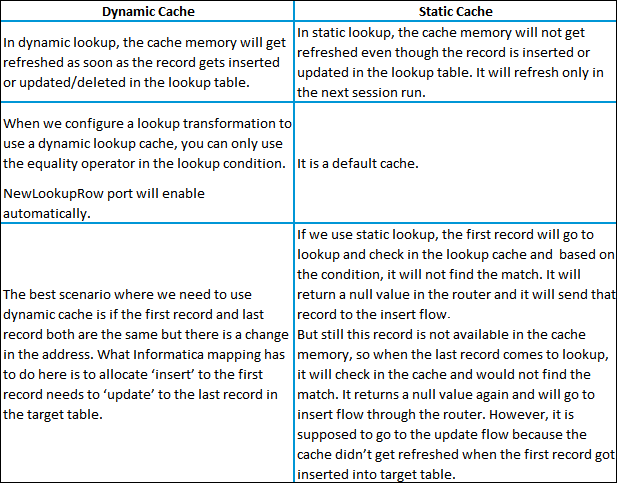
Active/Passive and Connected

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

* Connected lookup participates in the mapping(dataflow), just like any other transformation. Unconnected lookup is used when a lookup function is used instead in an expression transformation in the mapping in which case the lookup does not appear in the main flow (dataflow) of the mapping.
* Connected lookup can return more than one value(output port) whereas an Unconnected lookup gives only one output port. Unconnected lookups are reusable.

**When do we use dynamic cache and static cache in connected and unconnected lookup transformations?**

[**https://dwbi.org/etl/informatica/138-dynamic-lookup-cache**](https://dwbi.org/etl/informatica/138-dynamic-lookup-cache)



**How many types of transformations supported by sorted input?**

Aggregator Transformation

Joiner Transformation

Lookup Transformation support sorted input so that it will increase the session performance.

**Name the different lookup cache(s)?**

Informatica lookups can be cached or un-cached (no cache). Cached lookups can be either static or dynamic. A lookup cache can also be divided as persistent or non-persistent based on whether Informatica retains the cache even after completing session run or if it deletes it.

Static cache

Dynamic cache

Persistent cache

Shared cache

Recache

**Is ‘sorter’ an active or passive transformation?**

It is an active transformation because it removes the duplicates from the key and consequently changes the number of rows.

**Name 4 output files that Informatica server creates during session running:**

Session log

Workflow log

Errors log

Badfile

**What is the difference between stop and abort?**

Stop command immediately kills the reading process and doesn’t have any timeout period.

Abort command gives a time out period of 60 secs to the Informatica server to finish the dtm process else it kills the dtm process.

**G. What is Update Override? What are the differences between SQL Override and Update Override?**

Update Override – It is an option available in TARGET instance. By default, Target table is updated based on Primary key values. To update the Target table on non-primary key values, we can generate the default Query and override the Query according to the requirement. Suppose for example, if we want to update the record in target table, when a column value=’AAA’ then, we can include this condition in Where clause of default Query.

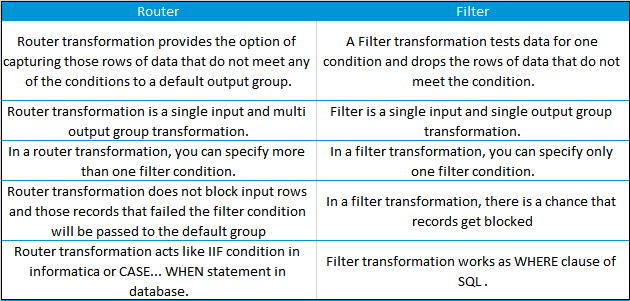
Coming to SQL override – It is an option available in Source Qualifier and Lookup transformation where we can include joins, filters, Group by and Order by.

Router and Filter Trasnformation

Router: Discard data to another group or default group if no conditions are met

Filter: Discard data based on condition

**Filter and Router**



**What are the various types of transformation?**

Aggregator transformation

Expression transformation

Filter transformation

Joiner transformation

Lookup transformation

Normalizer transformation

Rank transformation

Router transformation

Sequence generator transformation

Stored procedure transformation

Sorter transformation

Update strategy transformation

XML source qualifier transformation

**What is the difference between active and passive transformation?**

**Active Transformation**:- An active transformation can perform any of the following actions:

Change the number of rows that pass through the transformation: For instance, the Filter transformation is active because it removes rows that do not meet the filter condition.

Change the transaction boundary: For e.g., the Transaction Control transformation is active because it defines a commit or roll back transaction based on an expression evaluated for each row.

Change the row type: For e.g., the Update Strategy transformation is active because it flags rows for insert, delete, update, or reject.

**Passive Transformation**: A passive transformation is one which will satisfy all these conditions:

Does not change the number of rows that pass through the transformation

Maintains the transaction boundary

Maintains the row type

**Why is sorter an active transformation?**

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 that were compared during the sort operation. The number of input rows will vary as compared to the output rows and hence it is an active transformation.

**When do you use SQL override in a lookup transformation?**

You should override the lookup query in the following circumstances:

1. Override the ORDER BY clause. Create the ORDER BY clause with fewer columns to increase performance. When you override the ORDER BY clause, you must suppress the generated ORDER BY clause with a comment notation.  
   Note: If you use pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with a comment notation.
2. A lookup table name or column names contains a reserved word. If the table name or any column name in the lookup query contains a reserved word, you must ensure that they are enclosed in quotes.
3. Use parameters and variables. Use parameters and variables when you enter a lookup SQL override. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the SQL statement, or use a parameter or variable as the SQL query. For example, you can use a session parameter, $ParamMyLkpOverride, as the lookup SQL query, and set $ParamMyLkpOverride to the SQL statement in a parameter file. The designer cannot expand parameters and variables in the query override and does not validate it when you use a parameter or variable. The integration service expands the parameters and variables when you run the session.
4. A lookup column name contains a slash (/) character. When generating the default lookup query, the designer and integration service replace any slash character (/) in the lookup column name with an underscore character. To query lookup column names containing the slash character, override the default lookup query, replace the underscore characters with the slash character, and enclose the column name in double quotes.
5. Add a WHERE clause. Use a lookup SQL override to add a WHERE clause to the default SQL statement. You might want to use the WHERE clause to reduce the number of rows included in the cache. When you add a WHERE clause to a Lookup transformation using a dynamic cache, use a Filter transformation before the Lookup transformation to pass rows into the dynamic cache that match the WHERE clause.  
   Note: The session fails if you include large object ports in a WHERE clause.
6. Other. Use a lookup SQL override if you want to query lookup data from multiple lookups or if you want to modify the data queried from the lookup table before the Integration Service caches the lookup rows. For example, use TO\_CHAR to convert dates to strings.

**What are data driven sessions?**

When you configure a session using update strategy, the session property data driven instructs Informatica server to use the instructions coded in mapping to flag the rows for insert, update, delete or reject. This is done by mentioning DD\_UPDATE or DD\_INSERT or DD\_DELETE in the update strategy transformation.

**What are mapplets?**

A Mapplet is a reusable object that we create in the Mapplet Designer.

It contains a set of transformations and lets us reuse that transformation logic in multiple mappings.

**What is the use of source qualifier?**

The source qualifier transformation is an active, connected transformation used to represent the rows that the integrations service reads when it runs a session. **The source qualifier transformation converts the source data types to the Informatica native data types**. So, you should not alter the data types of the ports in the source qualifier transformation.

**What are the different ways to filter rows using Informatica transformations?**

Source Qualifier

Joiner

Filter

Router

**What are the different transformations where you can use a SQL override?**

Source Qualifier

Lookup

Target

**State the differences between SQL Override and Lookup Override?**

The role of SQL Override is to limit the number of incoming rows entering the mapping pipeline, whereas Lookup Override is used to limit the number of lookup rows to avoid the whole table scan by saving the lookup time and the cache it uses.

Lookup Override uses the “Order By” clause by default. SQL Override doesn’t use it and should be manually entered in the query if we require it

SQL Override can provide any kind of ‘join’ by writing the query

Lookup Override provides only Non-Equi joins.

Lookup Override gives only one record even if it finds multiple records for a single condition

SQL Override doesn’t do that.

**What are the different ways to implement parallel processing in Informatica?**

We can implement parallel processing using various types of partition algorithms:

**Database partitioning:** The Integration Service queries the database system for table partition information. It reads partitioned data from the corresponding nodes in the database.

**Round-Robin Partitioning:** Using this partitioning algorithm, the Integration service distributes data evenly among all partitions. It makes sense to use round-robin partitioning when you need to distribute rows evenly and do not need to group data among partitions.

**Hash Auto-Keys Partitioning:** The Powercenter Server uses a hash function to group rows of data among partitions. When the hash auto-key partition is used, the Integration Service uses all grouped or sorted ports as a compound partition key. You can use hash auto-keys partitioning at or before Rank, Sorter, and unsorted Aggregator transformations to ensure that rows are grouped properly before they enter these transformations.

**Hash User-Keys Partitioning:** Here, the Integration Service uses a hash function to group rows of data among partitions based on a user-defined partition key. You can individually choose the ports that define the partition key.

**Key Range Partitioning:** With this type of partitioning, you can specify one or more ports to form a compound partition key for a source or target. The Integration Service then passes data to each partition depending on the ranges you specify for each port.

**Pass-through Partitioning:** In this type of partitioning, the Integration Service passes all rows from one partition point to the next partition point without redistributing them.

**Explain shared cache and re cache.**

To answer this question, it is essential to understand persistence cache. If we are performing lookup on a table, it looks up all the data brings it inside the data cache. However, at the end of each session, the Informatica server deletes all the cache files. If you configure the lookup as a persistent cache, the server saves the lookup under an anonymous name. Shared cache allows you to use this cache in other mappings by directing it to an existing cache.

Mention a few design and development best practices for Informatica.

Mapping design tips:Standards – sticking to consistent standards is beneficial in the long run. This includes naming conventions, descriptions, environment settings, parameter files, documentation, among others.

Reusability – in order to react quickly to potential changes, use Informatica components like mapplets, worklets, and reusable transformations.

Scalability – when designing and developing mappings, it is a good practice to keep volumes in mind. This is caching, queries, partitioning, initial vs incremental loads.

Simplicity – it is recommended to create multiple mappings instead of few complex ones. Use Staging Area and try to keep the processing logic as clear and simple as possible.

Modularity – use the modular design technique (common error handling, reprocessing).

Mapping development best practices

Source Qualifier – use shortcuts, extract only the necessary data, limit read of columns and rows on source. Try to use the default query options (User Defined Join, Filter) instead of using SQL Query override which may impact database resources and make unable to use partitioning and push-down.

Expressions – use local variables to limit the amount of redundant calculations, avoid datatype conversions, reduce invoking external scripts (coding outside of Informatica), provide comments, use operators (||, +, /) instead of functions. Keep in mind that numeric operations are generally faster than string operations.

Filter – use the Filter transformation as close to the source as possible. If multiple filters need to be applied, usually it’s more efficient to replace them with Router.

Aggregator – use sorted input, also use as early (close to the source) as possible and filter the data before aggregating.

Joiner – try to join the data in Source Qualifier wherever possible, and avoid outer joins. It is good practice to use a source with fewer rows, such as a Master source.

## Lookup – relational lookup should only return ports that meet the condition. Call Unconnected Lookup in expression (IIF). Replace large lookup tables with joins whenever possible. Review the database objects and add indexes to database columns when possible. Use Cache Calculator in session to eliminate paging in lookup cache.

## ****Mention a few design and development best practices for Informatica.****

Mapping design tips: Standards – sticking to consistent standards is beneficial in the long run. This includes naming conventions, descriptions, environment settings, parameter files, documentation, among others.

Reusability – in order to react quickly to potential changes, use Informatica components like mapplets, worklets, and reusable transformations.

Scalability – when designing and developing mappings, it is a good practice to keep volumes in mind. This is caching, queries, partitioning, initial vs incremental loads.

Simplicity – it is recommended to create multiple mappings instead of few complex ones. Use Staging Area and try to keep the processing logic as clear and simple as possible.

Modularity – use the modular design technique (common error handling, reprocessing).

Mapping development best practices

Source Qualifier – use shortcuts, extract only the necessary data, limit read of columns and rows on source. Try to use the default query options (User Defined Join, Filter) instead of using SQL Query override which may impact database resources and make unable to use partitioning and push-down.

Expressions – use local variables to limit the amount of redundant calculations, avoid datatype conversions, reduce invoking external scripts (coding outside of Informatica), provide comments, use operators (||, +, /) instead of functions. Keep in mind that numeric operations are generally faster than string operations.

Filter – use the Filter transformation as close to the source as possible. If multiple filters need to be applied, usually it’s more efficient to replace them with Router.

Aggregator – use sorted input, also use as early (close to the source) as possible and filter the data before aggregating.

Joiner – try to join the data in Source Qualifier wherever possible, and avoid outer joins. It is good practice to use a source with fewer rows, such as a Master source.

Lookup – relational lookup should only return ports that meet the condition. Call Unconnected Lookup in expression (IIF). Replace large lookup tables with joins whenever possible. Review the database objects and add indexes to database columns when possible. Use Cache Calculator in session to eliminate paging in lookup cache.

<http://www.etltechblog.com/2012/05/tuning-sessions-for-better-performance.html>

**What are the differences between Source Qualifier and Joiner Transformation?**

The Source Qualifier can join data originating from the same source database. We can join two or more tables with primary key-foreign key relationships by linking the sources to one Source Qualifier transformation.

If we have a requirement to join the mid-stream or the sources are heterogeneous, then we will have to use the Joiner transformation to join the data.

**Differentiate between joiner and Lookup Transformation.**

Below are the differences between lookup and joiner transformation:

In lookup we can override the query but in joiner we cannot.

In lookup we can provide different types of operators like – “>,<,>=,<=,!=” but, in joiner only “= “ (equal to )operator is available.

In lookup we can restrict the number of rows while reading the relational table using lookup override but, in joiner we cannot restrict the number of rows while reading.

In joiner we can join the tables based on- Normal Join, Master Outer, Detail Outer and Full Outer Join but, in lookup this facility is not available .Lookup behaves like Left Outer Join of database.