I have done my B.sc Computers from KU University in 2009. Ap.After that I had an opportunity to work for a **Wipro Technologies** from Oct 2006 to Aug 2008 where I started off my career as an ETL developer. I have been with Wipro almost 2 years then I shifted to **Ness Technologies** In Aug 2008. Presently I am working with Ness

Total I have 3.5 Years of experience in DWH using informatica tool in development and Enhancement projects. Primarily I worked on healthcare and manufacturing domains.

In my Current project my roles & responsibilities are basically

- I am working with onsite offshore model so we use to get the tasks from my onsite team.
- As a developer first I need to understand the physical data model i.e dimensions and facts; their relationship & also functional specifications that tells the business requirement designed by Business Analyst.
- I involved into the preparation of source to target mapping sheet (tech Specs) which tell us what is the source and target and which column we need to map to target column and also what would be the business logic. This document gives the clear picture for the development.
- Creating informatica mappings, sessions and workflows using different transformations to implement business logic.
- Preparation of Unit test cases also one of my responsibilities as per the business requirement.
- And also involved into Unit testing for the mappings developed by myself.
- I use to source code review for the mappings & workflows developed by my team members.
- And also involved into the preparation of deployment plan which contains list of mappings and workflows they need to migrate based on this deployment team can migrate the code from one environment to another environment.
- Once the code rollout to production we also work with the production support team for 2 weeks where we parallel give the KT. So we also prepare the KT document as well for the production team.

**Coming to My Current Project:**

Currently I am working for XXX project for YYY client. Generally YYY does not have a manufacturing unit, What BIZ (Business) use to do here is before quarter ends they will call for quotations for primary supply channels this process we called as a RFQ's(Request for quotations).Once BIZ creates RFQ automatically notification will go to supply channels .So these supply channels send back their respective quoted values that we called it as response from the supply channel. After that biz will start negotiations with supply channels then they approve the RFQ's.

All these activities (Creating RFQ, supplier response and approve RFQ etc.)Performed in the oracle apps this is source frontend tool application. These data which get stored into OLTP. So the OLTP contains all the RFQs, supplier response and approval status data.

We have some Oracle jobs running between OLTP and ODS which replicate the OLTP data to ODS. It is designed in such a way that any transaction entering into the OLTP is immediately reflected into the ODS.

We have a staging area where we load the entire ODS data into staging tables for this we have created some ETL informatica mappings these mappings will truncate and reload the staging tables for each session run. Before loading to staging tables we are dropping indexes then after loading bulk data we are recreating indexes using store procedures.

Then we extract all this data from stage & load it into the dimensions & facts on top of dims and facts we have created some materialized views as per the report requirement .Finally report directly pulls the data from MV .These reports /dashboards performance always good because we are not doing any calculation at reporting level. These dashboards/reports can be used for the analysis purpose like say how many RFQs created, how many RFQs approved, how many RFQs got responded from the supply channels?

What is the budget?

What is budget approved?

Who is the approval manager pending with whom what is the feedback of the supply

channels from the past etc?

They don't have the BI design, so they are using the manual process to achieve the above by exporting the excel sheet; so we can do the drill up, drill down & get all the detailed reports by charts.

## ORACLE

**How strong you are in SQL& PL/SQL?**

1) I am good in SQL,I use to write the source qualifier queries for informatica mappings as per the business requirement.
2) I am comfortable to work with joins; co related queries, sub queries, analyzing tables, inline views and materialized views.
3) As a informatica developer I could not get more opportunity to work on pl/sql side. But I worked on PL/SQL to informatica migration project so I do have exposure on procedure, function and triggers.

**What is the difference between view and materialized view?**

| View | Materialized view |
|---|---|
| A view has a logical existence. It does not contain data. | A materialized view has a physical existence. |
| Its not a database object. | It is a database object. |
| We cannot perform DML operation on view. | We can perform DML operation on materialized view. |
| When we do select * from view it will fetch the data from base table. | When we do select * from materialized view it will fetch the data from materialized view. |
| In view we cannot schedule to refresh. | In materialized view we can schedule to refresh. |
| | We can keep aggregated data into materialized view. Materialized view can be created based on multiple tables. |

**Materialized View**

Materialized view is very essential for reporting. If we don't have the materialized view it will directly fetch the data from dimension and facts. This process is very slow since it involves multiple joins. So the same report logic if we put in the materialized view. We can fetch the data directly from materialized view for reporting purpose. So that we can avoid multiple joins at report run time.

It is always necessary to refresh the materialized view. Then it can simply perform select statement on materialized view.

**Difference between Trigger and Procedure**

| Triggers | Stored Procedures |
|---|---|
| In trigger no need to execute manually. Triggers will be fired automatically.<br>**Triggers** that run implicitly when an INSERT, UPDATE, or DELETE statement is issued against the associated table. | Where as in procedure we need to execute manually. |

**Differences between sub-query and co-related sub-query**

| Sub-query | Co-related sub-query |
|---|---|
| A sub-query is executed once for the parent Query | Where as co-related sub-query is executed once for each row of the parent query. |
| Example:<br>Select * from emp where deptno in (select deptno from dept); | Example:<br>Select a.* from emp e where sal >= (select avg(sal) from emp a where a.deptno=e.deptno group by a.deptno); |

**Differences between where clause and having clause**

| Where clause | Having clause |
|---|---|
| Both where and having clause can be used to filter the data. | |
| Where as in where clause it is not mandatory. | But having clause we need to use it with the group by. |
| Where clause applies to the individual rows. | Where as having clause is used to test some condition on the group rather than on individual rows. |
| Where clause is used to restrict rows. | But having clause is used to restrict groups. |
| Restrict normal query by where | Restrict group by function by having |
| In where clause every record is filtered based on where. | In having clause it is with aggregate records (group by functions). |

**Differences between stored procedure and functions**

| Stored Procedure | Functions |
|---|---|
| Stored procedure may or may not return values. | Function should return at least one output parameter. Can return more than one parameter using OUT argument. |
| Stored procedure can be used to solve the business logic. | Function can be used to calculations |
| Stored procedure is a pre-compiled statement. | But function is not a pre-compiled statement. |
| Stored procedure accepts more than one argument. | Whereas function does not accept arguments. |
| Stored procedures are mainly used to process the tasks. | Functions are mainly used to compute values |
| Cannot be invoked from SQL statements. E.g. SELECT | Can be invoked form SQL statements e.g. SELECT |
| Can affect the state of database using commit. | Cannot affect the state of database. |
| Stored as a pseudo-code in database i.e. | Parsed and compiled at runtime. |

| | |
|---|---|
| compiled form. | |

**Differences between rowid and rownum**

| Rowid | Rownum |
|---|---|
| Rowid is an oracle internal id that is allocated every time a new record is inserted in a table. This ID is unique and cannot be changed by the user. | Rownum is a row number returned by a select statement. |
| Rowid is permanent. | Rownum is temporary. |
| Rowid is a globally unique identifier for a row in a database. It is created at the time the row is inserted into the table, and destroyed when it is removed from a table. | The rownum pseudocoloumn returns a number indicating the order in which oracle selects the row from a table or set of joined rows. |

**How to find out duplicate records in table?**

Select empno, count (*) from EMP group by empno having count (*)>1;

**How to delete a duplicate records in a table?**

Delete from EMP where rowid not in (select max (rowid) from EMP group by empno);

**What is your tuning approach if SQL query taking long time? Or how do u tune SQL query?**

If query taking long time then First will run the query in Explain Plan, The explain plan process stores data in the PLAN_TABLE.

it will give us execution plan of the query like whether the query is using the relevant indexes on the joining columns or indexes to support the query are missing.

If joining columns doesn't have index then it will do the full table scan if it is full table scan the cost will be more then will create the indexes on the joining columns and will run the query it should give better performance and also needs to analyze the tables if analyzation happened long back. The ANALYZE statement can be used to gather statistics for a specific table, index or cluster using

ANALYZE TABLE employees COMPUTE STATISTICS;
If still have performance issue then will use HINTS, hint is nothing but a clue. We can use hints like

- ALL_ROWS

One of the hints that 'invokes' the Cost based optimizer
ALL_ROWS is usually used for *batch processing* or *data warehousing* systems.

(/*+ ALL_ROWS */)

- FIRST_ROWS
  One of the hints that 'invokes' the Cost based optimizer
  FIRST_ROWS is usually used for *OLTP* systems.

  (/*+ FIRST_ROWS */)
- CHOOSE
  One of the hints that 'invokes' the Cost based optimizer
  This hint lets the server choose (between ALL_ROWS and FIRST_ROWS, based on statistics gathered.
- HASH
  Hashes one table (full scan) and creates a hash index for that table. Then hashes other table and uses hash index to find corresponding records. Therefore not suitable for < or > join conditions.

  /*+ use_hash */

Hints are most useful to optimize the query performance.

## DWH Concepts

### Difference between OLTP and DWH/DS/OLAP

| OLTP | DWH/DSS/OLAP |
|---|---|
| OLTP maintains only current information. | OLAP contains full history. |
| It is a normalized structure. | It is a de-normalized structure. |
| Its volatile system. | Its non-volatile system. |
| It cannot be used for reporting purpose. | It's a pure reporting system. |
| Since it is normalized structure so here it requires multiple joins to fetch the data. | Here it does not require much joins to fetch the data. |
| It's not time variant. | Its time variant. |
| It's a pure relational model. | It's a dimensional model. |

### What is Staging area why we need it in DWH?

If target and source databases are different and target table volume is high it contains some millions of records in this scenario without staging table we need to design your informatica using look up to find out whether the record exists or not in the target table since target has huge volumes so its costly to create cache it will hit the performance.

If we create staging tables in the **target database** we can simply do outer join in the source qualifier to determine insert/update this approach will give you good performance.
It will avoid full table scan to determine insert/updates on target.

And also we can create index on staging tables since these tables were designed for specific application it will not impact to any other schemas/users.

While processing flat files to data warehousing we can perform **cleansing**

Data cleansing, also known as data scrubbing, is the process of ensuring that a set of data is correct and accurate. During data cleansing, records are checked for accuracy and consistency.

- Since it is one-to-one mapping from ODS to staging we do truncate and reload.
- We can create indexes in the staging state, to perform our source qualifier best.
- If we have the staging area no need to relay on the informatics transformation to known whether the record exists or not.

**ODS:**

My understanding of ODS is, its a replica of OLTP system and so the need of this, is to reduce the burden on production system (OLTP) while fetching data for loading targets. Hence its a mandate Requirement for every Warehouse.

So every day do we transfer data to ODS from OLTP to keep it up to date?
OLTP is a sensitive database they should not allow multiple select statements it may impact the performance as well as if something goes wrong while fetching data from OLTP to data warehouse it will directly impact the business.
ODS is the replication of OLTP.
ODS is usually getting refreshed through some oracle jobs.

**What is the difference between a primary key and a surrogate key?**

A **primary key** is a special constraint on a column or set of columns. A primary key constraint ensures that the column(s) so designated have no NULL values, and that every value is unique. Physically, a primary key is implemented by the database system using a unique index, and all the columns in the primary key must have been declared NOT NULL. A table may have only one primary key, but it may be composite (consist of more than one column).

A **surrogate key** is any column or set of columns that can be declared as the primary key *instead of* a "real" or **natural key**. Sometimes there can be several natural keys that could be declared as the primary key, and these are all called **candidate keys**. So a surrogate is a

candidate key. A table could actually **have more than one surrogate key**, although this would be unusual. The most common type of surrogate key is an incrementing integer, such as an auto increment column in MySQL, or a sequence in Oracle, or an identity column in SQL Server.

**Have you done any Performance tuning in informatica?**

1) Yes, One of my mapping was taking 3-4 hours to process 40 millions rows into staging table we don't have any transformation inside the mapping its 1 to 1 mapping .Here nothing is there to optimize the mapping so I created session partitions using key range on effective date column. It improved performance lot, rather than 4 hours it was running in 30 minutes for entire 40millions.Using partitions DTM will creates multiple reader and writer threads.
2) There was one more scenario where I got very good performance in the mapping level .Rather than using lookup transformation if we can able to do outer join in the source qualifier query override this will give you good performance if both lookup table and source were in the same database. If lookup tables is huge volumes then creating cache is costly.
3) And also if we can able to optimize mapping using less no of transformations always gives you good performance.
4) If any mapping taking long time to execute then first we need to look in to source and target statistics in the monitor for the throughput and also find out where exactly the bottle neck by looking busy percentage in the session log will come to know which transformation taking more time ,if your source query is the bottle neck then it will show in the end of the session log as "query issued to database "that means there is a performance issue in the source query.we need to tune the query using .

**How strong you are in UNIX?**

1) I have Unix shell scripting knowledge whatever informatica required like

If we want to run workflows in Unix using PMCMD.

Below is the script to run workflow using inix

cd /pmar/informatica/pc/pmserver/

/pmar/informatica/pc/pmserver/**pmcmd startworkflow -u $INFA_USER -p $INFA_PASSWD -s $INFA_SERVER:$INFA_PORT -f $INFA_FOLDER -wait $1** >> $LOG_PATH/$LOG_FILE

2) And if we suppose to process flat files using informatica but those files were exists in remote server then we have to write script to get ftp into informatica server before start process those files.

3) And also file watch mean that if indicator file available in the specified location then we need to start our informatica jobs otherwise will send email notification using

Mail X command saying that previous jobs didn't completed successfully something like that.

4) Using shell script update parameter file with session start time and end time.

This kind of scripting knowledge I do have. If any new UNIX requirement comes then I can Google and get the solution implement the same.

**What is use of Shortcuts in informatica?**

If we copy source definaltions or target definations or mapplets from Shared folder to any other folders that will become a shortcut.

Let's assume we have imported some source and target definitions in a shared folder after that we are using those sources and target definitions in another folders as a shortcut in some mappings.

If any modifications occur in the backend (Database) structure like adding new columns or drop existing columns either in source or target I f we reimport into shared folder those new changes automatically it would reflect in all folder/mappings wherever we used those sources or target definitions.

**How to do Dymanic File generation in Informatica?**

I want to generate the separate file for every employee (as per Name, it should generate file).It has to generate 5 flat files and name of the flat file is corresponding employee name that is the requirement.
Below is my mapping.

Source (Table) -> SQ -> Target (FF)

Source:

| Dept | Ename | EmpNo |
|---|---|---|
| A | S | 22 |
| A | R | 27 |
| B | P | 29 |
| B | X | 30 |
| B | U | 34 |

This functionality was added in informatica 8.5 onwards earlier versions it was not there.

We can achieve it with use of transaction control and special "FileName" port in the target file .

In order to generate the target file names from the mapping, we should make use of the special "FileName" port in the target file. You can't create this special port from the usual New port button. There is a special button with label "F" on it to the right most corner of the target flat file when viewed in "Target Designer".

When you have different sets of input data with different target files created, use the same instance, but with a Transaction Control transformation which defines the boundary for the source sets.

in target flat file there is option in column tab i.e filename as column.
when you click that one non editable column gets created in metadata of target.

in transaction control give condition **as iif(not isnull(emp_no),tc_commit_before,continue) else tc_commit_before**

map the emp_no column to target's filename column

ur mapping will be like this

source -> squlf-> transaction control-> target

run it ,separate files will be created by name of Ename

**How do u populate 1st record to 1st target , 2nd record to 2nd target ,3rd record to 3rd target and 4th record to 1st target through informatica?**

We can do using sequence generator by setting end value=3 and enable cycle option.then in the router take 3 goups

In 1st group specify condition as seq next value=1 pass those records to 1st target simillarly

In 2nd group specify condition as seq next value=2 pass those records to 2nd target

In 3rd group specify condition as seq next value=3 pass those records to 3rd target.

Since we have enabled cycle option after reaching end value sequence generator will start from 1,for the 4th record seq.next value is 1 so it will go to 1st target.

**How do you perform incremental logic or Delta or CDC?**

Incremental means suppose today we processed 100 records ,for tomorrow run u need to extract whatever the records inserted newly and updated after previous run based on last updated timestamp (Yesterday run) this process called as incremental or delta.

**Approach_1:** Using set max var ()

1) First need to create mapping var ($$Pre_sess_max_upd)and assign initial value as old date (01/01/1940).
2) Then override source qualifier query to fetch only LAT_UPD_DATE >=$$Pre_sess_max_upd (Mapping var)
3) In the expression assign max last_upd_date value to $$Pre_sess_max_upd(mapping var) using set max var
4) Because its var so it stores the max last upd_date value in the repository, in the next run our source qualifier query will fetch only the records updated or inseted after previous run.

**Approach_2:** Using parameter file

1 First need to create mapping parameter ($$Pre_sess_start_tmst )and assign initial value as old date (01/01/1940) in the parameterfile.
2 Then override source qualifier query to fetch only LAT_UPD_DATE >=$$Pre_sess_start_tmst (Mapping var)
3 Update mapping parameter($$Pre_sess_start_tmst) values in the parameter file using shell script or another mapping after first session get completed successfully
4 Because its mapping parameter so every time we need to update the value in the parameter file after comptetion of main session.

**Approach_3:** Using oracle Control tables

1. First we need to create two control tables cont_tbl_1 and cont_tbl_1 with structure of session_st_time,wf_name
2. Then insert one record in each table with session_st_time=1/1/1940 and workflow_name
3. create two store procedures one for update cont_tbl_1 with session st_time, set property of store procedure type as Source_pre_load .
4. In 2nd store procedure set property of store procedure type as Target _Post_load.this proc will update the session _st_time in Cont_tbl_2 from cnt_tbl_1.
5. Then override source qualifier query to fetch only LAT_UPD_DATE >=(Select session_st_time from cont_tbl_2 where workflow name='Actual work flow name'.

**Difference between dynamic lkp and static lkp cache?**

1. In Dynamic lkp the cache memory will get refreshed as soon as the record get inserted or updated/deleted in the lookup table where as in static lookup the cache memory will not get refreshed even though record inserted or updated in the lookup table it will refresh only in the next session run.
2. Best example where we need to use dynamic chache is if suppose first record and last record both are same but there is a change in the address what informatica mapping has to do here is first record needs to get insert and last record should get update in the target table.
3. If we use static look up first record it will go to lookup and check in the lkp cache based on the condition it will not find the macth so it returns null value then in the router will send that recod to insert flow.
4. But still this record does not available in the cach memory so when the last record comes to look up it will check in the cache it will not find the match it returns null values again it will go to insert flow through router but it suppose to go update flow because cache didn't refreshed when the first record get insert in to target table. So if we use dynamic look up we can achieve our requirement because first time record get insert then immediately cache also

get refresh with the target data. When we process last record it will find the match in the cache so it returns the value then router will route that record to update flow.

**What is the difference between snow flake and star schema**

| Star Schema | Snow Flake Schema |
|---|---|
| The star schema is the simplest data warehouse scheme. | Snowflake schema is a more complex data warehouse model than a star schema. |
| In star schema each of the dimensions is represented in a single table .It should not have any hierarchies between dims. | In snow flake schema at least one hierarchy should exists between dimension tables. |
| It contains a fact table surrounded by dimension tables. If the dimensions are de-normalized, we say it is a star schema design. | It contains a fact table surrounded by dimension tables. If a dimension is normalized, we say it is a snow flaked design. |
| In star schema only one join establishes the relationship between the fact table and any one of the dimension tables. | In snow flake schema since there is relationship between the dimensions tables it has to do many joins to fetch the data. |
| A star schema optimizes the performance by keeping queries simple and providing fast response time. All the information about the each level is stored in one row. | Snowflake schemas normalize dimensions to eliminated redundancy. The result is more complex queries and reduced query performance. |
| It is called a star schema because the diagram resembles a star. | It is called a snowflake schema because the diagram resembles a snowflake. |

**Difference between data mart and data warehouse**

| Data Mart | Data Warehouse |
|---|---|
| Data mart is usually sponsored at the department level and developed with a specific issue or subject in mind, a data mart is a data warehouse with a focused objective. | Data warehouse is a “Subject-Oriented, Integrated, Time-Variant, Nonvolatile collection of data in support of decision making”. |
| A data mart is used on a business division/ department level. | A data warehouse is used on an enterprise level |
| A Data Mart is a subset of data from a Data Warehouse. Data Marts are built for specific user groups. | A Data Warehouse is simply an integrated consolidation of data from a variety of sources that is specially designed to support strategic and tactical decision making. |
| By providing decision makers with only a subset of data from the Data Warehouse, Privacy, Performance and Clarity Objectives can be attained. | The main objective of Data Warehouse is to provide an integrated environment and coherent picture of the business at a point in time. |

**Differences between connected lookup and unconnected lookup**

| Connected Lookup | Unconnected Lookup |
|---|---|

| | |
|---|---|
| This is connected to pipleline and receives the input values from pipleline. | Which is not connected to pipeline and receives input values from the result of a: LKP expression in another transformation via arguments. |
| We cannot use this lookup more than once in a mapping. | We can use this transformation more than once within the mapping |
| We can return multiple columns from the same row. | Designate one return port (R), returns one column from each row. |
| We can configure to use dynamic cache. | We cannot configure to use dynamic cache. |
| Pass multiple output values to another transformation. Link lookup/output ports to another transformation. | Pass one output value to another transformation. The lookup/output/return port passes the value to the transformation calling: LKP expression. |
| Use a dynamic or static cache | Use a static cache |
| Supports user defined default values. | Does not support user defined default values. |
| Cache includes the lookup source column in the lookup condition and the lookup source columns that are output ports. | Cache includes all lookup/output ports in the lookup condition and the lookup/return port. |

**What is the difference between joiner and lookup**

| Joiner | Lookup |
|---|---|
| In joiner on multiple matches it will return all matching records. | In lookup it will return either first record or last record or any value or error value. |
| In joiner we cannot configure to use persistence cache, shared cache, uncached and dynamic cache | Where as in lookup we can configure to use persistence cache, shared cache, uncached and dynamic cache. |
| We cannot override the query in joiner | We can override the query in lookup to fetch the data from multiple tables. |
| We can perform outer join in joiner transformation. | We cannot perform outer join in lookup transformation. |
| We cannot use relational operators in joiner transformation.(i.e. <,>,<= and so on) | Where as in lookup we can use the relation operators. (i.e. <,>,<= and so on) |

**What is the difference between source qualifier and lookup**

| Source Qualifier | Lookup |
|---|---|
| In source qualifier it will push all the matching records. | Where as in lookup we can restrict whether to display first value, last value or any value |
| In source qualifier there is no concept of cache. | Where as in lookup we concentrate on cache concept. |
| When both source and lookup are in same database we can use source qualifier. | When the source and lookup table exists in different database then we need to use lookup. |

**Differences between dynamic lookup and static lookup**

| Dynamic Lookup Cache | Static Lookup Cache |
|---|---|
| In dynamic lookup the cache memory will get refreshed as soon as the record get inserted or updated/deleted in the lookup table. | In static lookup the cache memory will not get refreshed even though record inserted or updated in the lookup table it will refresh only in the next session run. |
| When we configure a lookup transformation to use a dynamic lookup cache, you can only use the equality operator in the lookup condition.<br>NewLookupRow port will enable automatically. | It is a default cache. |
| Best example where we need to use dynamic cache is if suppose first record and last record both are same but there is a change in the address. What informatica mapping has to do here is first record needs to get insert and last record should get update in the target table. | If we use static lookup first record it will go to lookup and check in the lookup cache based on the condition it will not find the match so it will return null value then in the router it will send that record to insert flow.<br>But still this record dose not available in the cache memory so when the last record comes to lookup it will check in the cache it will not find the match so it returns null value again it will go to insert flow through router but it is suppose to go to update flow because cache didn't get refreshed when the first record get inserted into target table. |

**How to Process multiple flat files to single target table through informatica if all files are same structure?**

We can process all flat files through one mapping and one session using list file.

First we need to create list file using unix script for all flat file the extension of the list file is .LST.

This list file it will have only flat file names.

At session level we need to set

source file directory as **list file path**

And source file name as **list file name**

And file type as **indirect.**

**SCD Type-II Effective-Date Approach**

- We have one of the dimension in current project called resource dimension. Here we are maintaining the history to keep track of SCD changes.
- To maintain the history in slowly changing dimension or resource dimension. We followed SCD Type-II Effective-Date approach.
- My resource dimension structure would be eff-start-date, eff-end-date, s.k and source columns.
- Whenever I do a insert into dimension I would populate eff-start-date with sysdate, eff-end-date with future date and s.k as a sequence number.
- If the record already present in my dimension but there is change in the source data. In that case what I need to do is
- Update the previous record eff-end-date with sysdate and insert as a new record with source data.

**Informatica design to implement SDC Type-II effective-date approach**

- Once you fetch the record from source qualifier. We will send it to lookup to find out whether the record is present in the target or not based on source primary key column.
- Once we find the match in the lookup we are taking SCD column and s.k column from lookup to expression transformation.
- In lookup transformation we need to override the lookup override query to fetch active records from the dimension while building the cache.
- In expression transformation I can compare source with lookup return data.
- If the source and target data is same then I can make a flag as 'S'.
- If the source and target data is different then I can make a flag as 'U'.
- If source data does not exists in the target that means lookup returns null value. I can flag it as 'I'.
- Based on the flag values in router I can route the data into insert and update flow.
- If flag='I' or 'U' I will pass it to insert flow.
- If flag='U' I will pass this record to eff-date update flow
- When we do insert we are passing the sequence value to s.k.
- Whenever we do update we are updating the eff-end-date column based on lookup return s.k value.

**Complex Mapping**

- We have one of the order file requirement. Requirement is every day in source system they will place filename with timestamp in informatica server.
- We have to process the same date file through informatica.
- Source file directory contain older than 30 days files with timestamps.
- For this requirement if I hardcode the timestamp for source file name it will process the same file every day.
- So what I did here is I created $InputFilename for source file name.

- Then I am going to use the parameter file to supply the values to session variables ($InputFilename).
- To update this parameter file I have created one more mapping.
- This mapping will update the parameter file with appended timestamp to file name.
- I make sure to run this parameter file update mapping before my actual mapping.

**How to handle errors in informatica?**

- We have one of the source with numerator and denominator values we need to calculate num/deno
- When populating to target.
- If deno=0 I should not load this record into target table.
- We need to send those records to flat file after completion of 1st session run. Shell script will check the file size.
- If the file size is greater than zero then it will send email notification to source system POC (point of contact) along with deno zero record file and appropriate email subject and body.
- If file size<=0 that means there is no records in flat file. In this case shell script will not send any email notification.
- Or
- We are expecting a not null value for one of the source column.
- If it is null that means it is a error record.
- We can use the above approach for error handling.

**Worklet**

Worklet is a set of reusable sessions. We cannot run the worklet without workflow.

If we want to run 2 workflow one after another.

- If both workflow exists in same folder we can create 2 worklet rather than creating 2 workfolws.
- Finally we can call these 2 worklets in one workflow.
- There we can set the dependency.
- If both workflows exists in different folders or repository then we cannot create worklet.
- We can set the dependency between these two workflow using shell script is one approach.
- The other approach is event wait and event rise.

**In shell script approach**

- As soon as it completes first workflow we are creating zero byte file (indicator file).
- If indicator file is available in particular location. We will run second workflow.
- If indicator file is not available we will wait for 5 minutes and again we will check for

the indicator. Like this we will continue the loop for 5 times i.e 30 minutes.

- After 30 minutes if the file does not exists we will send out email notification.

**Event wait and Event rise approach**

In event wait it will wait for infinite time. Till the indicator file is available.

**Why we need source qualifier?**
Simply it performs select statement.
Select statement fetches the data in the form of row.
Source qualifier will select the data from the source table.
It identifies the record from the source.

**Parameter file** it will supply the values to session level variables and mapping level variables.

**Variables are of two types:**

- Session level variables
- Mapping level variables

**Session level variables are of four types:**

- $DBConnection_Source
- $DBConnection_Target
- $InputFile
- $OutputFile

**Mapping level variables are of two types:**

- Variable
- Parameter

**What is the difference between mapping level and session level variables?**
Mapping level variables always starts with $$.
A session level variable always starts with $.

**Flat File**
Flat file is a collection of data in a file in the specific format.

Informatica can support two types of files

- Delimiter
- Fixed Width

In delimiter we need to specify the separator.

In fixed width we need to known about the format first. Means how many character to read for particular column.

In delimiter also it is necessary to know about the structure of the delimiter. Because to know about the headers.

If the file contains the header then in definition we need to skip the first row.

**List file:**

If you want to process multiple files with same structure. We don't need multiple mapping and multiple sessions.

We can use one mapping one session using list file option.

First we need to create the list file for all the files. Then we can use this file in the main mapping.

**Aggregator Transformation:**

**Transformation type:**
Active
Connected

The Aggregator transformation performs aggregate calculations, such as averages and sums. The Aggregator transformation is unlike the Expression transformation, in that you use the Aggregator transformation to perform calculations on groups. The Expression transformation permits you to perform calculations on a row-by-row basis only.

Components of the Aggregator Transformation:

The Aggregator is an active transformation, changing the number of rows in the pipeline. The Aggregator transformation has the following components and options

**Aggregate cache:** The Integration Service stores data in the aggregate cache until it completes aggregate calculations. It stores group values in an index cache and row data in the data cache.

**Aggregate expression:** Enter an expression in an output port. The expression can include non-aggregate expressions and conditional clauses.

**Group by port:** Indicate how to create groups. The port can be any input, input/output, output, or variable port. When grouping data, the Aggregator transformation outputs the last row of each group unless otherwise specified.

**Sorted input:** Select this option to improve session performance. To use sorted input, you must pass data to the Aggregator transformation sorted by group by port, in ascending or

descending order.

**Aggregate Expressions:**

The Designer allows aggregate expressions only in the Aggregator transformation. An aggregate expression can include conditional clauses and non-aggregate functions. It can also include one aggregate function nested within another aggregate function, such as:

MAX (COUNT (ITEM))

The result of an aggregate expression varies depending on the group by ports used in the transformation

**Aggregate Functions**

Use the following aggregate functions within an Aggregator transformation. You can nest one aggregate function within another aggregate function.

The transformation language includes the following aggregate functions:

AVG
COUNT

FIRST
LAST

MAX
MEDIAN

MIN
PERCENTILE

STDDEV
SUM

VARIANCE

When you use any of these functions, you must use them in an expression within an Aggregator transformation.

**Tips**

Use sorted input to decrease the use of aggregate caches.

Sorted input reduces the amount of data cached during the session and improves session performance. Use this option with the Sorter transformation to pass sorted data to the Aggregator transformation.

Limit connected input/output or output ports.

Limit the number of connected input/output or output ports to reduce the amount of data the Aggregator transformation stores in the data cache.

Filter the data before aggregating it.

If you use a Filter transformation in the mapping, place the transformation before the Aggregator transformation to reduce unnecessary aggregation.

**Normalizer Transformation:**

Transformation type:
Active
Connected

The Normalizer transformation receives a row that contains multiple-occurring columns and returns a row for each instance of the multiple-occurring data.

The Normalizer transformation parses multiple-occurring columns from COBOL sources, relational tables, or other sources. It can process multiple record types from a COBOL source that contains a REDEFINES clause.

The Normalizer transformation generates a key for each source row. The Integration Service increments the generated key sequence number each time it processes 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.

**SQL Transformation**

**Transformation type:**
Active/Passive
Connected

The SQL transformation processes SQL queries midstream in a pipeline. You can insert, delete, update, and retrieve rows from a database. You can pass the database connection information to the SQL transformation as input data at run time. The transformation processes external SQL scripts or SQL queries that you create in an SQL editor. The SQL transformation processes the query and returns rows and database errors.

For example, you might need to create database tables before adding new transactions. You can create an SQL transformation to create the tables in a workflow. The SQL transformation returns database errors in an output port. You can configure another workflow to run if the SQL transformation returns no errors.

When you create an SQL transformation, you configure the following options:

**Mode.** The SQL transformation runs in one of the following modes:

**Script mode.** The SQL transformation runs ANSI SQL scripts that are externally located. You pass a script name to the transformation with each input row. The SQL transformation outputs one row for each input row.

**Query mode.** The SQL transformation executes a query that you define in a query editor. You can pass strings or parameters to the query to define dynamic queries or change the selection parameters. You can output multiple rows when the query has a SELECT statement.

**Database type.** The type of database the SQL transformation connects to.

**Connection type.** Pass database connection information to the SQL transformation or use a connection object.

Script Mode

An SQL transformation running in script mode runs SQL scripts from text files. You pass each script file name from the source to the SQL transformation ScriptName port. The script file name contains the complete path to the script file.

When you configure the transformation to run in script mode, you create a passive transformation. The transformation returns one row for each input row. The output row contains results of the query and any database error.

When the SQL transformation runs in script mode, the query statement and query data do not change. When you need to run different queries in script mode, you pass the scripts in the source data. Use script mode to run data definition queries such as creating or dropping tables.

When you configure an SQL transformation to run in script mode, the Designer adds the ScriptName input port to the transformation

An SQL transformation configured for script mode has the following default ports:

| Port | Type | Description |
|---|---|---|
| ScriptName | Input | Receives the name of the script to execute for the current row. |
| ScriptResult | Output | Returns PASSED if the script execution succeeds for the row. Otherwise contains FAILED. |
| ScriptError | Output | Returns errors that occur when a script fails for a row. |

**Script Mode Rules and Guidelines**

- Use the following rules and guidelines for an SQL transformation that runs in script mode:
- You can use a static or dynamic database connection with script mode.
- To include multiple query statements in a script, you can separate them with a semicolon.
- You can use mapping variables or parameters in the script file name
- The script code page defaults to the locale of the operating system. You can change the locale of the script.
- You cannot use scripting languages such as Oracle PL/SQL or Microsoft/Sybase T-SQL in the script.
- You cannot use nested scripts where the SQL script calls another SQL script
- A script cannot accept run-time arguments.
- The script file must be accessible by the Integration Service. The Integration Service must have read permissions on the directory that contains the script. If the Integration Service uses operating system profiles, the operating system user of the operating system profile must have read permissions on the directory that contains the script.
- The Integration Service ignores the output of any SELECT statement you include in the SQL script. The SQL transformation in script mode does not output more than one row of data for each input row.

**Query Mode:**

When an SQL transformation runs in query mode, it executes an SQL query that you define in the transformation. You pass strings or parameters to the query from the transformation input ports to change the query statement or the query data.

When you configure the SQL transformation to run in query mode, you create an active transformation. The transformation can return multiple rows for each input row.

Create queries in the SQL transformation SQL Editor. To create a query, type the query statement in the SQL Editor main window. The SQL Editor provides a list of the transformation ports that you can reference in the query.

You can create the following types of SQL queries in the SQL transformation:

**Static SQL query**. The query statement does not change, but you can use query parameters to change the data. The Integration Service prepares the query once and runs the query for all input rows.

**Dynamic SQL query**. You can change the query statements and the data. The Integration Service prepares a query for each input row.
When you create a static query, the Integration Service prepares the SQL procedure once and executes it for each row. When you create a dynamic query, the Integration Service prepares the SQL for each input row. You can optimize performance by creating static queries.

**Query Mode Rules and Guidelines**

- Use the following rules and guidelines when you configure the SQL transformation to run in query mode:
- The number and the order of the output ports must match the number and order of the fields in the query SELECT clause.
- The native datatype of an output port in the transformation must match the datatype of the corresponding column in the database. The Integration Service generates a row error when the datatypes do not match.
- When the SQL query contains an INSERT, UPDATE, or DELETE clause, the transformation returns data to the SQLError port, the pass-through ports, and the NumRowsAffected port when it is enabled. If you add output ports the ports receive NULL data values.
- When the SQL query contains a SELECT statement and the transformation has a pass-through port, the transformation returns data to the pass-through port whether or not the query returns database data. The SQL transformation returns a row with NULL data in the output ports.
- You cannot add the "_output" suffix to output port names that you create.
- You cannot use the pass-through port to return data from a SELECT query.
- When the number of output ports is more than the number of columns in the SELECT clause, the extra ports receive a NULL value.
- When the number of output ports is less than the number of columns in the SELECT clause, the Integration Service generates a row error.
- You can use string substitution instead of parameter binding in a query. However, the input ports must be string datatypes.

**Java Transformation Overview**

**Transformation type:**

Active/Passive
Connected

The Java transformation provides a simple native programming interface to define transformation functionality with the Java programming language. You can use the Java transformation to quickly define simple or moderately complex transformation functionality without advanced knowledge of the Java programming language or an external Java development environment.

For example, you can define transformation logic to loop through input rows and generate multiple output rows based on a specific condition. You can also use expressions, user-defined functions, unconnected transformations, and mapping variables in the Java code.

**Transaction Control Transformation**

**Transformation type:**
Active
Connected

PowerCenter lets you control commit and roll back transactions based on a set of rows that pass through a Transaction Control transformation. A transaction is the set of rows bound by commit or roll back rows. You can define a transaction based on a varying number of input rows. You might want to define transactions based on a group of rows ordered on a common key, such as employee ID or order entry date.

In PowerCenter, you define transaction control at the following levels:
**Within a mapping.** Within a mapping, you use the Transaction Control transformation to define a transaction. You define transactions using an expression in a Transaction Control transformation. Based on the return value of the expression, you can choose to commit, roll back, or continue without any transaction changes.
**Within a session.** When you configure a session, you configure it for user-defined commit. You can choose to commit or roll back a transaction if the Integration Service fails to transform or write any row to the target.

When you run the session, the Integration Service evaluates the expression for each row that enters the transformation. When it evaluates a commit row, it commits all rows in the transaction to the target or targets. When the Integration Service evaluates a roll back row, it rolls back all rows in the transaction from the target or targets.

If the mapping has a flat file target you can generate an output file each time the Integration Service starts a new transaction. You can dynamically name each target flat file.