International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 12, December 2014 ISSN 2319 - 4847

ETL Transformations Performance Optimization

Sunil Kumar, PMP[1], Dr. M.P. Thapliyal[2] and Dr. Harish Chaudhary[3]

[1] Research Scholar at Department Of Computer Science and Engineering, Bhagwant University, Ajmer, India

[2] Department of Computer Science HNB Garhwal University, Srinagar Uttaranchal, India

[3] Dean & Professor (R& D) Rama University, Kanpur, Uttar Pradesh

ABSTRACT

The ETL data load process is mainly depends on how effectively the ETL transformations are used and configured properly at appropriate place. The proper usage of transformation and with proper setting will always give the best data transformation and loading into respective staging area. A transformation is the defined as object that generate, modify or pass the data to next stage of process. The transformation performs specific function in ETL mapping. The Integration Services (IS) will perform the required task based on type of transformation used in mapping. The transformations are divided into active and passive types of two categories only. The transformation may be connected, unconnected in mapping and mapplets. Generally an unconnected transformation is called within another transformation, and returns a value to that transformation then that returned values are used in subsequent transformations. In ETL data load process, mapping-level optimization takes time to implement, but can significantly boost performance. Sometimes the mapping is the biggest bottleneck in the load process due to business rules that determine the number and complexity of transformations in a mapping. Before deciding on the best route to optimize the mapping architecture, we need to resolve some basic issues. Tuning mappings is a tiered and iterative process. Some factors to consider when choosing tuning processes at the mapping level include the specific environment, software/ hardware limitations, and the number of records going through a mapping. If there is a change in number of rows that pass through the transformation, changes in rows type, change the transaction boundary then this kind of transformations are called active transformations. If a transformation does not change the number of rows that pass through the transformation, maintains the transaction boundary, and maintains the row type then these are called as passive transformation. The use of too many lookups and aggregators will slow down performance because each requires index cache and data cache to process the data. Since both are fighting for memory space, decreasing the number of these transformations in a mapping can help improve performance. Splitting this kind of mappings is another option to gain the performance. Limiting the number of Aggregators in a mapping will also give performance gain because a high number of Aggregators can increase I/O activity on the cache directory of Integration Services unless the seek/access time is fast on the directory itself, having too many aggregators can also cause a bottleneck. Similarly, too many Lookups in a mapping causes contention of disk and memory, that can lead to thrashing, leaving insufficient memory to run a mapping efficiently and smoothly.

Keywords: ETL (Extract, Transform, Load), ETL Transformation, Session and Workflow Optimization, Infrastructure , Application and Database Server Optimization.

1. INTRODUCTION

ETL process optimization includes various transformation optimization available in ETL tool supplied by tool vendor , The Informatica PowerCenter supports the XML Source Qualifier, XML Parser, XML Generator, Update Strategy, Unstructured Data, Transaction Control, Union, SQL, Stored Procedure, Source Qualifier, Sorter, Sequence Generator, Router, Rank, Output, Inputs, Normalizer, Lookup, Joiner, Java, HTTP, Filter, External Procedure, Expression, Data Masking, Custom, Application Source Qualifier, Aggregator transformations. Based on business purpose and requirements any transformations can be used in ETL mapping. Transformations may contain a combination of input, output, input/output, and variable port types. The Aggregator, Joiner, Rank, and Lookup transformations used the Cache memory for index and data cache files to process and transform the data. Minimizing and limiting connected input/output or output ports can reduce the amount of data the transformations store in the data cache and index cache files. The lookup transformations are used to bring the additional data related to a row from another database Table, Views, Synonyms, flat file sources. There are two types of lookup transformation called as Connected and unconnected lookup transformations. A connected Lookup transformation receives source data, performs a lookup, and returns data to the pipeline. An unconnected Lookup transformation is not connected to a source or target. A transformation in the pipeline calls the Lookup transformation with a :LKP expression. The unconnected Lookup transformation returns one column to the calling transformation. This transformation may be active/passive. The lookup transformation will return the result to target table or another transformation used in mapping. Lookup is used for following purposes:

- Find the corresponding values from another source.
- Find the multiple values from another source.
- To determine the rows exist in targets or not.
- Retrieve the value from another source and perform the business rule calculation.

2. LOOKUP TRANSFORMATION OPTIMIZATION:

The Integration Services must query, sort and compare values in the lookup condition columns. As a result, indexes on the database table should include every column used in a lookup condition. This can improve performance for both cached and un-cached lookups. In the case of a cached lookup, an ORDER BY condition is issued in the SQL statement used to create the cache files. Columns used in the ORDER BY condition should be indexed. The session log will contain the ORDER BY statement. In the case of an un-cached lookup, since a SQL statement created for each row passing into the lookup transformation, performance can be improved by indexing columns used in the lookup condition. If the volume of data in lookup source is not very high then caching the lookup transformation will improve the performance. The lookup transformation creates **Index cache** to hold the values from all ports which are part of lookup condition, **Data cache** hold the values from all output ports which are not part of the lookup condition. After cache is loaded values from lookup input ports that are part of lookup condition are compared with the index cache and when a match is found then rows from cache are included in the stream.

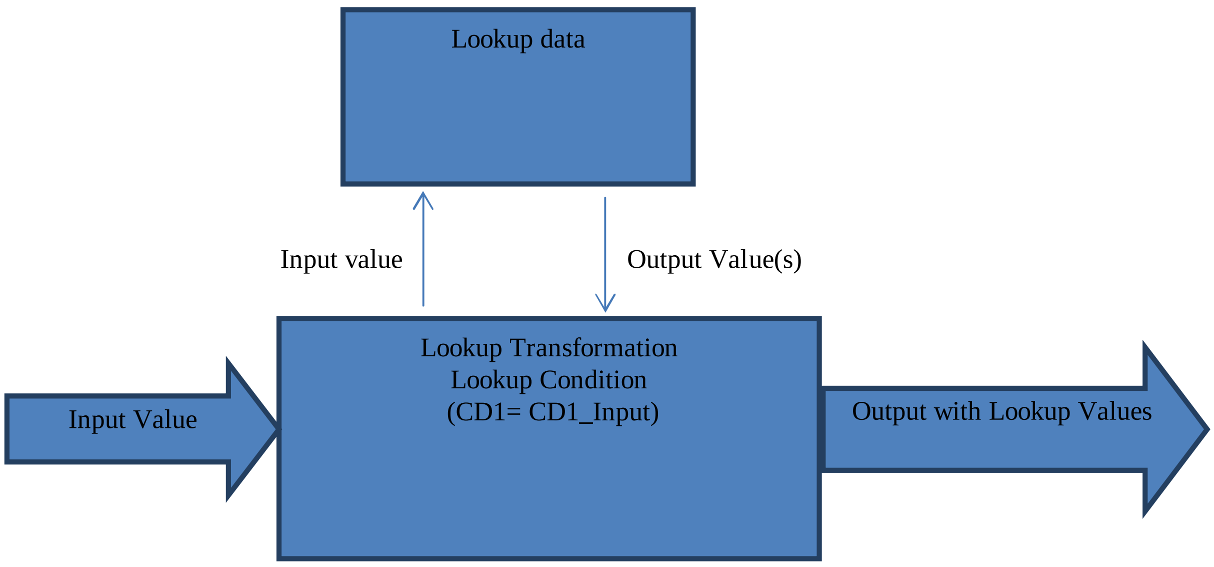

Figure 1 Lookup Transformation Process

Static and dynamic cache options can be used with lookup transformation. The static cache does not change during session run while dynamic cache Integration services can insert or update the rows in cache. The lookup transformation marks each rows for Insert or Update into target. The integration services cache enable option is available with Relational, Flat File and pipeline lookup. The Integration Service always caches the flat file lookups and the pipeline lookups. Also make sure to define the lookup policy on multiple matches from one of the following possible options.

- Integration service will report an error and does not return a row (report Error).
- Return the first row that matches the lookup condition (Use first value).
- Return the last rows that match the lookup condition (Use last value).
- Return the all matching rows (Use all values) .
- The Integration Service returns the first value that matches the lookup condition. It creates an index based on the key ports instead of all Lookup transformation ports (use any value).

Enabling lookup cache option cached the lookup table on server and will give the performance boost with relational lookup tables because one local SQL SELECT statement is needed. If the lookup source data is not changes frequently then enabling lookup cache persistent will give the faster performance. The persistent cache stores the data in Server hard drive and can be used by next session. Persistent cache can improve the performance but stale data may create the major problem in production environment. The sorted input option increases lookup performance for file lookups.

Enabling Pre-build Lookup Cache option allows the Integration Service to build the lookup cache before the Lookup transformation receives the data. The Integration Service can build multiple lookup cache files at the same time to improve performance. We can configure this option in the mapping or the session. The Integration Service uses the session-level setting if we configure the Lookup transformation option as Auto. Configure one of the following three options for pre build lookup cache:

- **Auto:** The Integration Service uses the value configured in the session.
- **Always allowed:** The Integration Service can build the lookup cache before the Lookup transformation receives the first source row. The Integration Service creates an additional pipeline to build the cache.
- **Always disallowed:** The Integration Service cannot build the lookup cache before the Lookup transformation receives the first row.

Configure the number of pipelines that the Integration Service can build concurrently. Configure the Additional Concurrent Pipelines for Lookup Cache Creation session property. The Integration Service can pre-build lookup cache if this property is greater than zero. Generally lookup table should be cached if table size is approximately 300 MB of memory. The caching the small /mid-size table will improve the query performance during session execution. The cache should not be enabled for big data tables consisting of hundreds of millions of records. Lookup tables must be considered for caches if table meet one of following conditions criteria to determine if a lookup should be cached:

- Design and Code the lookup table into the ETL mapping. Select a standard set of data from the source by writing SQL Query in Source Qualifier to load sample 100681 rows of source data.
- Run the Session/mapping with caching turned off and save the log file.
- Run the mapping with caching turned on and save the log to a new name than the log created in previous step.
- Look in the cached lookup log and determine how long it takes to cache the lookup object and note this time in seconds i.e. Lookup Time in Seconds **(LTS)**
- In the non-cached log, take the time from the last lookup cache to the end of the load in seconds and divide it into the number or rows being processed on Cached Rows Per Second **(NRS)**
- In the cached log, take the time from the last lookup cache to the end of the load in seconds and divide it into number of rows being processed i.e. Cached Rows Per Seconds **(CRS)**
- Use the following formula to find the breakeven row point X:
  X = [LTS*NRS*CRS]/ [CRS-NRS]
- If we expected source records are less than X, it is better to not cache the lookup. If we expected source records are more than X, it is better to cache the lookup.
- If we expected source records are less than X, it is better to not cache the lookup. If we expected source records are more than X, it is better to cache the lookup.
- For example: Assume the lookup takes 180 seconds to cache (LTS=180). Assume with a cached lookup the load is 232 rows per second (CRS=300). Assume with a non-cached lookup the load is 177 rows per second (NRS = 177). The formula X = [LTS*NRS*CRS]/[CRS-NRS] = (180*177*300)/(300-177) = 77707 Rows , Thus, if the source has less than 77707 records, the lookup should not be cached. If it has more than 77707 records, then the lookup should be cached.

3. FILTER & ROUTER TRANSFORMATION OPTIMIZATION:

Filtering data as early as possible in the data flow improves the efficiency of a mapping. Instead of using a Filter Transformation to remove a sizeable number of rows in the middle or end of a mapping, use a filter on the Source Qualifier or a Filter Transformation immediately after the source qualifier to improve performance. Filter transformations are most effective when a simple integer or TRUE/FALSE expression is used in the filter condition. As best practice , avoid complex expressions when creating the filter condition. Filters and routers should also be used to drop rejected rows from an Update Strategy transformation if rejected rows do not need to be saved for business purposes. Replace multiple filter transformations with a router transformation. This reduces the number of transformations in the mapping and makes the mapping easier to follow and manage.

4. AGGREGATOR TRANSFORMATIONS OPTIMIZATION:

Aggregator Transformations often slow performance because they must group data before processing it. Use simple columns in the group by condition to make the Aggregator Transformation more efficient. When possible, use numbers instead of strings or dates in the GROUP BY columns. Also avoid complex expressions in the Aggregator expressions, especially in GROUP BY ports. Use the Sorted Input option in the aggregator. This option requires that data sent to the aggregator be sorted in the order in which the ports are used in the aggregators group by. The Sorted Input option decreases the use of aggregate caches. When it is used, the Integration Services assumes all data is sorted by group and, as a group is passed through an aggregator, calculations can be performed and information passed on to the next transformation. Without sorted input, the Server must wait for all rows of data before processing aggregate calculations. Use of the Sorted Inputs option is usually accompanied by a Source Qualifier which uses the Number of Sorted Ports option. Use an Expression and Update Strategy instead of an Aggregator Transformation. This technique can only be used if the source data can be sorted. Further, using this option assumes that a mapping is using an Aggregator with Sorted Input option. In the Expression Transformation, the use of variable ports is required to hold data from the previous row of data processed. The premise is to use the previous row of data to determine whether the current row is a part of the current group or is the beginning of a new group. Thus, if the row is a part of the current group, then its data would be used to continue calculating the current group function.

5. JOINER TRANSFORMATIONS OPTIMIZATION:

Joiner transformations can slow performance because they need additional space in memory at run time to hold intermediate result data set. Define the rows from the smaller set of data in the joiner as the Master rows. The Master rows are cached to memory and the detail records are then compared to rows in the cache of the Master rows. In order to minimize memory requirements, the smaller set of data should be cached and thus set as Master. Use Normal joins whenever possible. Normal joins are faster than outer joins and the resulting set of data is also smaller. Use the database to do the join when sourcing data from the same database schema. Database systems usually can perform the join more quickly than the Integration Services, so a SQL override or a join condition should be used when joining multiple tables from the same database schema.

6. SEQUENCE GENERATOR TRANSFORMATIONS OPTIMIZATION:

Sequence Generator transformations need to determine the next available sequence number, thus increasing the Number of Cached Values property can increase performance. This property determines the number of values the Integration Services caches at one time. If it is set to cache no values then the Integration Services must query the Informatica repository each time to determine what the next number which can be used is. Configuring the Number of Cached Values to a value greater than 1000 should be considered. It should be noted any cached values not used in the course of a session are lost since the sequence generator value in the repository is set, when it is called next time, to give the next set of cache values.

7. EXTERNAL PROCEDURE TRANSFORMATIONS OPTIMIZATION:

For the most part, making calls to external procedures slows down a session. If possible, avoid the use of these Transformations, which include Stored Procedures, External Procedures and Advanced External Procedures.

8. EXPRESSION TRANSFORMATION OPTIMIZATION:

Processing field level transformations takes time. If the transformation expressions are complex, then processing will be more slower. It's often possible to get a 10- 25% performance improvement by optimizing complex field level transformations. Use the target table mapping reports or the Metadata Reporter to examine the transformations. Likely candidates for optimization are the fields with the most complex expressions. Keep in mind that there may be more than one field causing performance problems so each field with complex expression need to be tested for performance gain. Expressions transformation must be optimized for the following manner to troubleshoot the performance bottlenecks.

- Run the session/mapping with the original expression and capture the total session execution time.
- Copy the mapping and replace half the complex expressions with constant value then run the session/mapping and capture the total session execution time.
- Make another copy of the mapping and replace the other half of the remaining complex expressions with a constant value then run the session/mapping and capture the total session execution time.
- Factoring out common logic can reduce the number of times a mapping performs the same logic. If a mapping performs the same logic multiple times in a mapping, moving the task upstream in the mapping may allow the logic to be done just once. For example, a mapping has five target tables. Each target requires

a Social Security Number or Tax Identification Number lookup. Instead of performing the lookup right before each target, move the lookup to a position before the data flow splits.

- Anytime a function is called it takes database resources to process. Therefore functions call need to be minimized. There are several common scenarios where function calls can be reduced or eliminated.

- Aggregate function calls can sometime be reduced. In the case of each aggregate function call, the Integration Services must search and group the data. Thus the following expression: SUM(Col A) + SUM(Col B) Can be optimized to Expression SUM(Col A + Col B)

- In general, operators are faster than functions, so operators should be used whenever possible. For example if we have an expression which involves a CONCAT function such as: CONCAT(CONCAT(FIRST_NAME,), LAST_NAME) It can be optimized to FIRST_NAME || || LAST_NAME by using Operator || in mapping.

- Remember that IIF() is a function that returns a value, not just a logical test. This allows many logical statements to be written in a more compact and easy manner. For example:

```
IIF(FLAG_A=Y and FLAG_B=Y and FLAG_C=Y, VAL_A+VAL_B+VAL_C,

IIF(FLAG_A=Y and FLAG_B=Y and FLAG_C=N, VAL_A+VAL_B,

IIF(FLAG_A=Y and FLAG_B=N and FLAG_C=Y, VAL_A+VAL_C,

IIF(FLAG_A=Y and FLAG_B=N and FLAG_C=N, VAL_A,

IIF(FLAG_A=N and FLAG_B=Y and FLAG_C=Y, VAL_B+VAL_C,

IIF(FLAG_A=N and FLAG_B=Y and FLAG_C=N, VAL_B,

IIF(FLAG_A=N and FLAG_B=N and FLAG_C=Y, VAL_C,

IIF(FLAG_A=N and FLAG_B=N and FLAG_C=N, 0.0))))))))
```

Above logical expression can be optimized to as follows to by using IIF function.

```
IIF(FLAG_A=Y, VAL_A, 0.0) + IIF(FLAG_B=Y, VAL_B, 0.0) + IIF(FLAG_C=Y, VAL_C, 0.0)
```

The original expression had 8 IIFs, 16 ANDs and 24 comparisons. The optimized expression results in 3 IIFs, 3 comparisons and 2 additions to perform the calculation.

- Be creative in making expressions more efficient. The following is an example of rework of an expression which eliminates three comparisons down to one: For example: IIF(X=1 OR X=5 OR X=9, 'yes', 'no') can be optimized to IIF(MOD(X, 4) = 1, 'yes', 'no') in summary calculate once, Use Many Times

- Avoid calculating or testing the same value multiple times. If the same sub-expression is used several times in a transformation, consider making the sub-expression a local variable. The local variable can be used only within the transformation but by calculating the variable only once can speed performance.

- Choose Numeric versus String Operations, The Integration Services processes numeric operations faster than string operations. For example, if a lookup is done on a large amount of data on two columns, EMPLOYEE_NAME and EMPLOYEE_ID, configuring the lookup around EMPLOYEE_ID improves performance.

- **Optimizing Char-Char and Char-Varchar Comparisons:** When the Integration Services performs comparisons between CHAR and VARCHAR columns, it slows each time it finds trailing blank spaces in the row. The Treat CHAR as CHAR On Read option can be set in the Integration Services setup so that the Integration Services does not trim trailing spaces from the end of CHAR source fields.

- **Use DECODE instead of LOOKUP :** When a LOOKUP function is used, the Integration Services must lookup a table in the database. When a DECODE function is used, the lookup values are incorporated into the expression itself so the Integration Services does not need to lookup a separate table. Thus, when looking up a small set of unchanging values, using DECODE may improve performance.

- **Reduce the Number of Trans**formatio**ns in a Mapping:** Whenever possible the number of transformations should be reduced. As there is always overhead involved in moving data between transformations. Along the same lines, unnecessary links between transformations should be removed to minimize the amount of data moved. This is especially important with data being pulled from the Source Qualifier Transformation.

- **Use Pre- and Post-Session SQL Commands:** Specify pre- and post-session SQL commands in the properties tab of the Source Qualifier transformation and in the properties tab of the target instance in a mapping. To increase the load speed, use these commands to drop indexes on the target before the session runs then recreate them when the session completes. Use any command that is valid for the database type. However, the PowerCenter Server does not allow nested comments, even though the database might support.

- Use mapping parameters and mapping variables in SQL executed against the source, but not against the target.

- Use a semi-colon (;) to separate multiple statements.

- The PowerCenter Server ignores semi-colons within single quotes, double quotes, or within /* ...*/. If we need to use a semi-colon outside of quotes or comments, we can escape it with a back slash (\). The Workflow Manager does not validate the SQL.

9. USE ENVIRONMENTAL SQL

For relational databases execute SQL commands in the database environment when connecting to the database. This can be used for source, target, lookup, and stored procedure connection. We can set isolation levels on the source and target systems to avoid deadlocks. Follow the guidelines mentioned above for using the SQL statements.

10. CONCLUSION:

The performance goal must be balanced between application tools, underlying database and used hardware component. Allow each component to perform their own task independently. In lookup tables, delete all unused columns and keep only the fields/ports that are used in the mapping. If possible, replace lookups by joiner transformation or single source qualifier transformation. The Joiner transformation takes more time than source qualifier transformation. If lookup transformation specifies several conditions, then place conditions that use equality operator '=' first in the conditions that appear in the conditions tab then uses the other conditions. In the SQL override query of the lookup table, there will be an ORDER BY clause. Remove it if not needed or put fewer column names in the ORDER BY list. Do not use caching Option If Source is small and lookup table is large and lookup is done on the primary key of the lookup table. Cache the lookup table columns definitely If lookup table is small and source is large. If lookup data is static, use persistent cache. Persistent caches help to save and reuse cache files. If several sessions in the same job use the same lookup table, then using persistent cache will help the sessions to reuse cache files. In case of static lookups, cache files will be built from memory cache instead of from the database, which will improve the performance. If source is huge and lookup table is also huge, then also use persistent cache. If target table is the lookup table, then use dynamic cache. The Integration Services updates the lookup cache as it passes rows to the target. Use only the lookups we want in the mapping. Too many lookups inside a mapping will slow down the session. If lookup table has a lot of data, then it will take too long to cache or fit in memory. So move those fields to source qualifier and then join with the main table. If there are several lookups with the same data set, then share the caches. If we are going to return only 1 row, then use unconnected lookup. All data are read into cache in the order the fields are listed in lookup ports. If we have an index that is even partially in this order, the loading of these lookups can be speeded up. If the table that we use for look up has an index (or if we have privilege to add index to the table in the database, do so), then the performance would increase both for cached and un cached lookups. Utilize the database server for sorting , aggregation and grouping high volume of data in database layer instead of application layer , also use the staging tables for intermediate data processing and transformations and localize the all target tables in same database schema instead of multiple database schemas. Do not use database based sequence generators because this requires additional store procedure call as a result it will degrade the performance by factor of three times and turn off the verbose logging and collect performance statistics property in session to improve the performance.

References

[1] https://community.informatica.com/thread/39270

[2] http://www.itap.purdue.edu/ea/files/informatica_tuning_guide.doc

[3] http://www.orafaq.com/wiki/Oracle_database_Performance_Tuning_FAQ

[4] http://www.kimballgroup.com/category/business-intelligence-and-data-warehouse-articles/

[5] http://www.dwbiconcepts.com/etl/14-etl-informatica/8-all-about-informatica-lookup.html