Performance Tuning Techniques for Handling High Volume of Data in Informatica
Performance Tuning Techniques for Handling High Volume of Data in Informatica

Table of Contents
1. Overview .................................................................................................................. 03
2. Scenario .................................................................................................................... 04
3. Architectural Overview ............................................................................................ 05
4. Steps involved in Performance Tuning ................................................................. 05
   4.1 Database Level Tuning ....................................................................................... 06
   4.2 Informatica Level Tuning .................................................................................. 08
   4.3 Balancing the Constraints ................................................................................. 13
5. Iterative Testing ........................................................................................................ 15
   5.1 Test Results ........................................................................................................ 15
1. Overview

Data is - and has been from the beginning - created, stored and retrieved by disparate, incompatible systems. Between 30% and 35% of all the data in the industry is still on mainframes, in languages and data structures that are archaic and generally unavailable. The wave of specialty applications like HR, Sales, Accounting, ERP, Manufacturing etc, have all contributed their share to the chaos. The latest (and perhaps most disruptive) development is the growth in outsourcing, in all kinds and flavors, including business processes, IT management and geographic, adding even more complexity to the mix.

When it comes to data integration, there are three perspectives that matter. First, business users need an accurate and holistic view of all their information. Second, IT management needs to do more with less even though the data volumes are increasing dramatically and finally, IT developers need to reduce time to results.

Informatica provides the market's leading data integration platform. ETL Mappings are designed for data loading into the data warehouse environment for better reporting purposes which in turn helps understand the business trends in a much better way. The major problem faced by anyone working with Informatica ETL is to design a mapping(s) which does not compromise on its performance. But most of the time we end up creating a mapping which achieves only the functionality but suffers in terms of performance.

This paper will walk you through the process of achieving good performance improvement in designing new Informatica mapping and also fine tune the performance of existing Informatica ETL loads. After going through this paper you will have fair knowledge on the performance break points that are to be considered while designing the mapping or fine tuning the existing ones.

In general, Informatica works in synchronization with its sources. In order to achieve good performance, tuning has to be carried out at two levels namely

DB Level Tuning

To achieve good performance, we first need to ensure that all the bottlenecks on the DB side are removed by which we can ensure that the sources are in sync and hence full utilization of the source system is achieved. The below table (Table1) gives you the details of how to overcome the DB side bottlenecks and the results achieved.

<table>
<thead>
<tr>
<th>S.No</th>
<th>How?</th>
<th>Why?</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check for the INDEXES on Key columns</td>
<td>The index on a Key field in a table adds a key factor to the overall performance while retrieving data from the table.</td>
<td>Normal index applied on all Key columns significantly improved the performance of the ETL Load.</td>
</tr>
<tr>
<td>2</td>
<td>Prepare an INDEX analysis Document</td>
<td>INDEX analysis document is prepared to keep track of the different indexes on different key columns</td>
<td>Decision on usage and application of INDEX on needed columns is arrived.</td>
</tr>
<tr>
<td>3</td>
<td>Use 'Explain Plan' for all the Source Qualifier Queries</td>
<td>Explain Plan executes your query and records the “plan” that Oracle devises to execute your query</td>
<td>Provides a way of writing better Source Qualifier Queries and modifying existing ones</td>
</tr>
</tbody>
</table>

Informatica level Tuning

Once the DB level tunings are completed and now all the sources are in sync and fully tuned, we can start looking at the Informatica level tuning. The below table (Table2) gives you the details of how to overcome the Informatica side bottlenecks and the results achieved.

<table>
<thead>
<tr>
<th>S.No</th>
<th>How?</th>
<th>Why?</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Removal of unwanted fields</td>
<td>The Source Qualifier Query should have a select statement with only the fields that are required and which get loaded into the Target table.</td>
<td>This will bring down the running time of the SQ Queries and this means the data will be fetched quickly from the database</td>
</tr>
</tbody>
</table>
Performance Tuning Techniques for Handling High Volume of Data in Informatica

2. Scenario

This whitepaper will take an example of an application namely Service Contracts. This application had huge performance issues having its load running for over 48 hrs and still not completing. The idea was to improve the performance of the load by fine tuning the mappings and the SQ queries involved in the load and thereby removing any bottlenecks present to ensure a smooth completion of the Load in an optimal time.

This will be achieved by trying various combinations of DB and Transformation changes made to the mapping and by iterative testing. The performance tuning methods mentioned below can be applied to virtually any kind of application which suffers performance issues.

<table>
<thead>
<tr>
<th>S.No</th>
<th>How?</th>
<th>Why?</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Avoid contraints in WHERE Clause</td>
<td>The usage of WHERE… IN… clause in the SQ Query should be avoided since it consumes more time in the query completion.</td>
<td>Instead use WHERE… EXISTS… clause which will results in better performance.</td>
</tr>
<tr>
<td>3</td>
<td>LOOK UP &amp; FILTER transformation</td>
<td>The Lookup and Filter transformations combination always work very effectively in order to achieve good performance while handling huge data loads in the ETL process.</td>
<td>This result is faster load operations and hence the performance improves.</td>
</tr>
</tbody>
</table>

Table 1

Also we need to establish a balance between these two levels by way of shifting the constraints according to the need and adopt a right method for testing the performance. Here we adopt the Rapid Evaluation and Iterative Testing (RITE) method for testing the performance improvement.

As per Production Session logs

<table>
<thead>
<tr>
<th>Gr.</th>
<th>Session Name</th>
<th>Start Time</th>
<th>End Time</th>
<th>Approx. Time</th>
<th>No. of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>s_m_upd_P_Dw_Process_Control_Lcg</td>
<td>Sun Sep 02 06:54:13 2007</td>
<td>Mon Sep 03 02:52:07 2007</td>
<td>20 hrs</td>
<td>257099</td>
</tr>
<tr>
<td>2</td>
<td>s_m_dd_Vldate_CT_IB_TRANS_STG</td>
<td>Fri Aug 31 20:05:34 2007</td>
<td>Sat Sep 01 08:04:34 2007</td>
<td>12 hrs</td>
<td>528747</td>
</tr>
<tr>
<td>3</td>
<td>s_m_dd_Vldate_CT_IB_TRANS_STG</td>
<td>Mon Sep 03 02:53:10 2007</td>
<td>Mon Sep 03 03:31:07 2007</td>
<td>30 min</td>
<td>291350</td>
</tr>
<tr>
<td>4</td>
<td>s_m_upd_B_E_Fct Dw Process Cont</td>
<td>Sat Sep 01 08:04:46 2007</td>
<td>Sun Sep 02 01:11:05 2007</td>
<td>16 hrs</td>
<td>257099</td>
</tr>
<tr>
<td>5</td>
<td>s_m_upd_C_Fct Dw Process Co</td>
<td>Sun Sep 02 01:11:40 2007</td>
<td>Sun Sep 02 04:50:00 2007</td>
<td>4 hrs</td>
<td>271688</td>
</tr>
<tr>
<td>6</td>
<td>s_m_del_CT_IB_TRANS_STG</td>
<td>Sun Sep 02 04:50:18 2007</td>
<td>Sun Sep 02 06:50:34 2007</td>
<td>2 hrs</td>
<td>528747</td>
</tr>
</tbody>
</table>
3. Architectural Overview

The architecture of the Load process is shown below.

The Load Process

- ERP pushes data into the STG table (CT_IB_TRANS_STG) with a particular submission number and a corresponding entry into the DWPCL for the same submission number with the PROJECT_STATUS as NULL.
- All the data from the ERROR table (CT_IB_TRANS_ERR) is copied back to the STG table and deleted from the ERR table. If the submission number of the ERROR table records is greater than that of the data from ERP, then the error records are processed first.
- ‘P’ records are created in the DWPCL for the submission numbers of both old and new submissions.
- Partitions are created in the Back up table CT_IB_TRANS_BAK based on the PERIOD
  - The current month partition along with previous 3 months partitions are retained in the back up table and rest all partitions are dropped.
- Data in the STG table is pushed to the CT_IB_TRANS_BAK table before the validation process as back up data.
- In the STG to STG validation, the following validations are carried out
  - After the validation process the STG table will have the DW_PROC_STATUS column updated to either ‘G’ or ‘B’ for all the records
  - Even if a single record is Bad, i.e. DW_PROC_STATUS is ‘B’, all the records in that batch are updated as Bad records
  - The DWPCL is updated with PROJECT_STATUS as ‘E’ with appropriate DW_ERROR_MSG for the current submission and submissions greater than the current one.
  - If all records are good, then they are loaded into the FCT
- All the bad records are inserted into the ERR table.
- All the good records are loaded into the following FCT tables
  - CT_IB_FCT - Insert/Update
  - CT_CONTRACT_LINES_FCT - Insert/Update
  - CT_BILLINGS_FCT - Only Insert

The DWPCL is then closed appropriately.

- All the records in the STG table are deleted for this particular load.

4. Steps involved in Performance Tuning

There are two levels involved in achieving significant performance improvement while handling huge data volumes during the ETL Load process. The following are the levels that need to be followed.
4.1 Database Level Tuning

The architecture of the Load process is shown below

1. Check for the INDEXES on Key columns

The index on a Key field in a table adds a key factor to the overall performance while retrieving data from the table. There are different types of indexes available in Oracle and here we will discuss about the usage of BITMAP index. Bitmap indexes are widely used in data warehousing environments. The environments typically have large amounts of data and ad hoc queries but a low level of concurrent DML transactions.

How does it work?

The bitmap index stores the column values in bits. Each bit represents a single value. For example, the DW_PROC_STATUS column has two possible values: ‘G’ and ‘B’. Three bit will be used in the bitmap to capture the index on the DW_PROC_STATUS column (refer Figure1). So the more distinct the value, more space is required to store the bitmap.

When to use it?

- **Low cardinality**
  
  The BITMAP index works best when there is a low cardinality of data.
  
  - If the number of distinct values of a column is less than 1% of the number of rows in the table or if the values in a column are repeated more than 100 times, then the column is a candidate for a bitmap index.
  
  - B-tree indexes are most effective for high-cardinality data: i.e., data with many possible values, such as CUSTOMER_NAME or PHONE_NUMBER.

- **No or little insert/update**

  Updating bitmap indexes take a lot of resources. Building and maintaining an index structure can be expensive and can consume resources such as disk space, CPU, and I/O capacity. Designers must ensure that the benefits of any index outweigh the negatives of index maintenance.

  Use this simple estimation guide for the cost of index maintenance: each index maintained by an INSERT, DELETE, or UPDATE of the indexed keys requires about three times as much resource as the actual DML operation on the table. What this means is that if you INSERT into a table with three indexes, then it will be approximately 10 times slower than an INSERT into a table with no indexes. For DML and particularly for INSERT-heavy applications, the index design should be seriously reviewed, which might require a compromise between the query and INSERT performance.

In the current scenario there is a BITMAP index on the DW_PROC_STATUS column in one of the tables. This field either stores ‘G’ or ‘B’ only and this value is updated based on the validity of the data. Although the cardinality of data was less with respect to the huge volume of data, the Insert/Update on the column was by extent very large and this brings down the performance drastically. The BITMAP index best work for retrieving the data from DB than for updating the field.

To over come this, the Index is dropped from DW_PROC_STATUS column and normal index is applied on all Key columns which significantly improved the performance of the ETL Load.
2. Prepare an INDEX analysis Document

Once the INDEX information has been gathered, all the detail that has been correlated and an INDEX analysis document is prepared to keep track of the different indexes on different key columns. Based on this information we can arrive at a solution on which column the index should be dropped and which column needs an index.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Index</th>
<th>Type</th>
<th>Index Fields</th>
<th>Partitions</th>
<th>Not Null Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT_IB_TRANS_STG</td>
<td>CT_IB_TRANS_STG_PROCSTS_BMI</td>
<td>Normal</td>
<td>DW_PROC_STATUS&lt;br&gt;COMPANY_CODE&lt;br&gt;PERIOD&lt;br&gt;INPUT_SOURCE_ID&lt;br&gt;SUBMISSION_MBR</td>
<td>Non-Partitioned</td>
<td>COMPANY_CODE&lt;br&gt;PERIOD&lt;br&gt;INPUT_SOURCE_ID&lt;br&gt;SUBMISSION_MBR&lt;br&gt;PRODUCT_NUMBER&lt;br&gt;DW_SOURCE</td>
</tr>
<tr>
<td>CT_IB_TRANS_ERR</td>
<td>CT_IB_TRANS_ERR_SUB_I</td>
<td>Normal</td>
<td>COMPANY_CODE&lt;br&gt;PERIOD&lt;br&gt;INPUT_SOURCE_ID&lt;br&gt;SUBMISSION_MBR</td>
<td>Non-Partitioned</td>
<td>COMPANY_CODE&lt;br&gt;PERIOD&lt;br&gt;INPUT_SOURCE_ID&lt;br&gt;PRODUCT_NUMBER&lt;br&gt;DW_SOURCE</td>
</tr>
<tr>
<td>CT_IB_TRANS_BAK</td>
<td>CT_IB_TRANS_BAK_BMI_I</td>
<td>Setup</td>
<td>COMPANY_CODE&lt;br&gt;PERIOD&lt;br&gt;INPUT_SOURCE_ID&lt;br&gt;SUBMISSION_MBR</td>
<td>Partitioned 10</td>
<td>No constraints</td>
</tr>
</tbody>
</table>

3. Use 'Explain Plan' for all the Source Qualifier Queries

Explain Plan is a great way to tune your queries. You will learn more about how the DBMS works “behind the scenes”, enabling you to write efficient queries the first time around. Explain Plan executes your query and records the “plan” that Oracle devises to execute your query. By examining this plan, you can find out if Oracle is picking the right indexes and joining your tables in the most efficient manner.

The EXPLAIN PLAN statement displays execution plans chosen by the Oracle optimizer for SELECT, UPDATE, INSERT and DELETE statements. A statement's execution plan is the sequence of operations Oracle performs to execute the statement. The components of execution plans include:

- An ordering of the tables referenced by the statement
- An access method for each table mentioned in the statement
- A join method for tables affected by join operations in the statement

The EXPLAIN PLAN output shows how Oracle executes SQL statements (refer Figure2). The EXPLAIN PLAN results alone, however, cannot differentiate between well-tuned statements and those that perform poorly. For example, if the EXPLAIN PLAN output shows that a statement uses an index, this does not mean the statement runs efficiently. Sometimes using indexes can be extremely inefficient. It is thus best to use EXPLAIN PLAN to determine an access plan and later prove it is the optimal plan through testing. It also gives you the information on the CPU utilization and the amount of resource it takes for execution of the query.

Figure 2
In the current scenario, the entire SQ Queries execution plan has been checked and they are modified to improve on its performance.

4.2 Informatica Level Tuning

1. Removal of unwanted fields

In general when a new mapping is created, all the fields from the source tables are selected, even though some of the fields might not be required to be pulled because they are not loaded into the target table. This becomes an overhead and hence this Source Qualifier (SQ) Query runs for a longer period of time and thereby degrades the performance.

In order to overcome this issue, the SQ Query should have a select statement with only the fields that are required and which get loaded into the Target table. This will bring down the running time of the SQ Queries and this means the data will be fetched quickly from the database and projected to the subsequent transformations.

In the current scenario the source table had 100+ fields and the SQ Query was fetching all the fields including those that were not getting loaded into the target table. (refer Figure 3,4)

The solution to improve on performance in this scenario was that, all the unwanted fields that were not required to be fetched and those that were not getting loaded into the Target table were removed from the SQ Query and hence the running time of the query was subsequently reduced and hence the bottleneck on its performance was removed. (Refer Figure 5, 6)
2. Avoid constraints in WHERE Clause

The usage of WHERE… IN… clause in the SQ Query should be avoided since it consumes more time in the query completion. Instead one should use the WHERE… EXISTS… clause which will result in better performance. This will remove the bottleneck on the query getting hung up because of the WHERE condition. Below is a simple example on how it works.

Eg: If you wish to retrieve from the department table, all the department numbers that have at least one employee assigned to them, write the query as:

```sql
SELECT deptno, deptname
FROM dept
WHERE deptno IN (SELECT deptno
FROM emp);
```

This query will run a full table scan on both the emp and dept tables. Even if there was an index on the deptno column of emp table, the sub-query would not be using the index hence, performance would suffer.

We can rewrite the query as:

```sql
SELECT deptno, deptname
FROM dept
WHERE EXISTS (SELECT deptno
FROM emp
WHERE dept.deptno = emp.deptno);
```

This query uses an existing deptno index on the emp table, making the query much faster. Thus, wherever possible use EXISTS in place of IN clause in a query.

In the current scenario, all the SQ Queries which had poor performance because of the use of WHERE… IN… clause (refer figure 7) in the query has been modified to use WHERE… EXISTS… clause (refer figure 8). Below mentioned is one of the scenarios in which we modified the query.

Figure 7

```sql
SELECT DISTINCT ct_ib_trans_stg.ROWID, ct_ib_trans_stg.submission_nbr,
    ct_ib_trans_stg.company_code, ct_ib_trans_stg.contract_nbr,
    ct_ib_trans_stg.input_source_id, ct_ib_trans_stg.period,
    ct_ib_trans_stg.dw_error_message,
    ct_ib_trans_stg.dw_proc_status, ct_ib_trans_stg.dw_source,
    ct_ib_trans_stg.modifier
FROM ct_ib_trans_stg
WHERE (company_code, input_source_id, period) IN (SELECT DISTINCT s.company_code, s.input_source_id,
    s.period
    FROM ct_ib_trans_stg s
    WHERE s.dw_proc_status = 'B'
    AND ct_ib_trans_stg.submission_nbr >= s.submission_nbr)
```

Figure 8

```sql
SELECT ct_ib_trans_stg.ROWID, ct_ib_trans_stg.submission_nbr,
    ct_ib_trans_stg.company_code, ct_ib_trans_stg.contract_nbr,
    ct_ib_trans_stg.input_source_id, ct_ib_trans_stg.period,
    ct_ib_trans_stg.dw_error_message, ct_ib_trans_stg.dw_proc_status,
    ct_ib_trans_stg.dw_source, ct_ib_trans_stg.modifier
FROM ct_ib_trans_stg
WHERE EXISTS (SELECT DISTINCT s.company_code, s.input_source_id, s.period
    FROM ct_ib_trans_stg s
    WHERE s.dw_proc_status = 'B'
    AND ct_ib_trans_stg.company_code = s.company_code
    AND ct_ib_trans_stg.input_source_id = s.input_source_id
    AND ct_ib_trans_stg.period = s.period
    AND ct_ib_trans_stg.submission_nbr >= s.submission_nbr)
```
3. LOOK UP & FILTER transformation
The Lookup and Filter transformations combination can be used very effectively in order to achieve good performance while handling huge data loads in the ETL process. The Lookup works best if we use an INDEXED key column for comparison in the lookup condition. The filter transformation placed after the lookup, filters the records based on the required condition and hence the FILTER transformation works well with this combination. In the current scenario, the Lookup-Filter combination was not used in many instances. (Refer figure 9, 10, 11).
These transformations were added wherever necessary without disturbing the existing functionality and the SQ query in the corresponding pipelines are modified accordingly. (Refer figure 12, 13, 14 and 15).
### 4.3 Balancing the Constraints

In order to achieve good performance in handling huge volumes of data, there should be a balance between the load on the database and on Informatica. The typical bottleneck is in the usage of a sorter. The sorter transformation can handle data only based on its cache size. The maximum size that the sorter can have is 2GB. But for optimal functioning of the SORTER transformation the recommended size is 8Mb. This cache cannot work well while handling huge volumes of data. Depending on the requirement the cache size can be changed and for optimal performance one can use the Formulae to calculate the cache size of Sorter as mentioned below:

\[
\text{# input rows} \times \left( \text{? column size} + 16 \right)
\]

In the current scenario, the volume of data was more than and the SORTER size was set to 8Mb, because of this the corresponding mapping session failed since the sorter size was much less to handle the huge data. (Refer figure 16 and 17).
To overcome this bottleneck, it is better to avoid usage of SORTER and this constraint is shifted to be handled at the database level. The SORTER transformation is removed and an ORDER BY clause is used at the SQL Query itself and hence the sorting of data happens at the source level itself. This improved the overall performance of data loading into the Target tables. (Refer figure 18 and 19).

Figure 18
5. Iterative Testing
Rapid Evaluation and Iterative Testing (RITE) is used as the testing method here to see if the performance has been improved or not. Initially the load is tested with minor changes and checked if there is any significant change in its performance. By this method we can revert and make the next set of changes and test the load again quicker than any other testing methods. (Refer Figure 20).

5.1 Test Results
In the current scenario, the load completely loaded data into the Target table in less than 90min.

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Session Name</th>
<th>Start Time</th>
<th>End Time</th>
<th>Approx. Time taken</th>
<th>No. of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>s_m_ins_Udp_C_FCT DwProcess_Control Log</td>
<td>Mon Oct 01 14:14:41 2007</td>
<td>Mon Oct 01 13:34:05 2007</td>
<td>21 min 49 sec</td>
<td>648129</td>
</tr>
</tbody>
</table>

This significant improvement in performance has been achieved by following the above said methods for performance tuning. These methods can be followed for any mapping designs.
Performance Tuning Techniques for Handling High Volume of Data in Informatica

Address
1095 Cranbury South River Road, Suite 10, Jamesburg, NJ 08831. Main: 609-409-6950 | Fax: 609-409-6910

Safe Harbor
Certain statements on this whitepaper concerning our future growth prospects are forward-looking statements, which involve a number of risks, and uncertainties that could cause actual results to differ materially from those in such forward-looking statements. The risks and uncertainties relating to these statements include, but are not limited to, risks and uncertainties regarding fluctuations in earnings, our ability to manage growth, intense competition in IT services including those factors which may affect our cost advantage, wage increases in India, our ability to attract and retain highly skilled professionals, time and cost overruns on fixed-price, fixed-time frame contracts, client concentration, restrictions on immigration, our ability to manage our international operations, reduced demand for technology in our key focus areas, disruptions in telecommunication networks, our ability to successfully complete and integrate potential acquisitions, liability for damages on our service contracts, the success of the companies in which Hexaware has made strategic investments, withdrawal of governmental fiscal incentives, political instability, legal restrictions on raising capital or acquiring companies outside India, and unauthorized use of our intellectual property and general economic conditions affecting our industry.