ETL Process Optimization & Performance Tuning
- A Case Study

Introduction
Our client has multiple ETL processes scheduled in a typical 24 hours day. Some ETL processes are dependent upon others. A new business change triggered process optimization and performance improvement project due to these process dependencies. Under this project, we studied existing ETL system, came up with various possible solutions, and implemented the best solution. As the part of this case study, one typical enhancement solution that we implemented for a daily process that loads 25 million records in each of 9 target tables is described below.

About Client
One among Fortune 50 companies.

Client’s Problem
- Potential business impact due to slow and long running ETL process.
- Long hours at work by production support professionals. Wastage of time in fire-fighting.
- Regular process failures due to process dependencies and timing conflicts.

Client Requirements
- Decrease total ETL process execution time from 560 minutes to 280 minutes, i.e., by 50%, for the process discussed in this case study.

Our Challenges
- Improve ETL performance
  - with minimal code change
  - with no change in IT operations
  - with no additional hardware & software cost

...slowness in the ETL process was causing delayed start of execution of downstream process & programs in 24 hours execution cycle…
…ultimately threatening business impacts and potential revenue loss…

50% improvement was required to contain the problem…our challenge was to improve performance of the ETL process with minimal and only code change.
Pre-Change Architecture

Following is the process flow before system was changed.

PROCESS FLOW (BEFORE CHANGE)

Start

Polar to start Process

INFORMATICA PROCESS

Group-1
- Pre-SQL
  - Table: T1
    - 4-Sessions: S1, S2, S3, S4
  - Table: T2
    - 4-Sessions: S1, S2, S3, S4
  - Table: T3
    - 4-Sessions: S1, S2, S3, S4

Group-2
- Pre-SQL
  - Table: T4
    - 4-Sessions: S1, S2, S3, S4
  - Table: T5
    - 4-Sessions: S1, S2, S3, S4
  - Table: T6
    - 4-Sessions: S1, S2, S3, S4

Group-3
- Pre-SQL
  - Table: T7
    - 4-Sessions: S1, S2, S3, S4
  - Table: T8
    - 4-Sessions: S1, S2, S3, S4
  - Table: T9
    - 4-Sessions: S1, S2, S3, S4

Pre-SQL: Truncate all tables in the group

End
Analysis and Findings

- There were multiple source database and source schemas.
- There were multiple complex data transformations in informatica before data loading.
- This daily process loads 25 million of records in each target table with average row length of 250 bytes.
- There were 3 groups of target tables and each group had 3 target tables. So, total 9 target tables were loaded by the daily process.
- Target tables in one group were loaded sequentially. Groups were executed in parallel.
- Regrouping tables doesn’t improve total execution time.
- Each target table worklet had 4 sessions running in sequence.
- All target tables of a group were truncated before start of informatica process for the group. No other DDL was executed on target tables.
- Even though, first three sessions were ready to run, whole process used to start late just because prerequisite for fourth session is not ready.
- Some downstream processes dependent on only first three sessions where getting delayed because worklet consists of all four sessions.
- Fourth session run and finish very quick (30 minutes)
- There were multiple costly lookups in informatica worklet for 4 out of 9 tables.
- More than 50% of the lookups can be eliminated and can be part of base query.
- Moving lookup to base query increase performance drastically (30% to 70% depending upon nature and complexity of lookup).

Opportunities Identified

- Move lookups to base query.
- Have 5 groups of target tables. Four groups with two tables each and one group with one table.
- Break current process into two independent processes where first process to have 3 sessions and second process to have one session.
- Drop indexes and disable constraints before start of data load and recreate indexes and enable constraints after end of data load.

…primary reason of performance problem was bad design of informatica workflow and its bad scheduling in 24 hours window…existing system violated the basic principles of ETL process…

…all target tables were truncated before start of load process and data load was being performed without dropping indexes and disabling constraints…

…unnecessary use of lookups even when they can be part of base query made process slower by at least 30%...
Implementation

We did not move lookup to base query because of huge code impact and because of extensive functional test requirement. This change is recommended for future as this also increases performance by at least 30%. We used override query feature of informatica to conclude this.

We did not increase number of groups of target tables due to additional infrastructure dependency.

We implemented last two opportunities listed above.

1. We broke current process into two independent processes with first process having 3 sessions and second process having one session. This eliminated the wait time before fourth session. Fourth session takes short time but it is functionally independent and can be started later. This helped starting some downstream jobs, which are dependent on first 3 sessions, earlier.

   As indicated above, we broke existing process into two. Post-SQLs of first process were used to create indexes and enable table constraints. We introduced an additional polar process before second informatica process to make sure Post-SQL activity of first process is complete. This was required to avoid start of second informatica process while the table is locked by first informatica process.

   This new implementation eliminated the hold of downstream jobs with dependency only on first three sessions. Now, such jobs can be executed immediately after first informatica process is completed.

2. We created APIs to drop indexes and disable constraints before start of data load and recreate indexes and enable constraints after the end of data load. Informatica process calls these APIs in pre-SQL (synchronous) and post-SQL (asynchronous). This increased performance by 74%.

Technologies Utilized

- Oracle database 11g Release 2 (with Partition option)
- Informatica 8

Final Outcome

Client’s requirement fulfilled and project delivered.

1. Performance improved by 74%:

<table>
<thead>
<tr>
<th>Measure</th>
<th>Existing</th>
<th>Target</th>
<th>Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total execution time</td>
<td>560 min</td>
<td>280 min</td>
<td>146 min</td>
</tr>
<tr>
<td>Reduction in execution</td>
<td>50%</td>
<td>74%</td>
<td></td>
</tr>
</tbody>
</table>

2. ETL Process optimized and time conflicts removed.
Post-Change Architecture

Following is high level process flow after system was changed.

![High Level Process Flow Diagram](image-url)
FIRST INFORMATICA PROCESS FLOW (AFTER CHANGE)

FIRST INFORMATICA PROCESS

Start

Pre-SQL

Table: T1

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T2

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T3

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T4

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T5

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T6

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T7

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T8

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

Table: T9

3-Sessions: S1, S2, S3

Pre-SQL

Post-SQL

End
Recommendations for Further Improvement

We made following recommendations for further improvement in performance and better handling of exceptions:

1. Move lookup to base query. This will increase performance by at least 30%.

2. Increase number of groups of target tables. This will increase performance by at least 25%. But this will need additional infrastructure.

3. Create day-wise partitions and session wise sub-partitions in database table. Day wise partitions will help running some of critical business processes even if today’s ETL not completed due to some reason. Sub-partitions will help in session level re-do. Currently, re-do of whole process is required in case of any error. Local indexes on day-wise partition recommended as per functional use.

…we created proof-of-concept to demonstrate the advantage of table and index partitions and sub-partitions for better availability & high performance of such ETL systems…
We created proof-of-concept and demonstrated above. Implementing these recommendations will require additional infrastructure, a complete cycle of rigorous technical & functional regression testing.

**Lessons Learned**

The key lessons learned were:

1. Presence of index and constraints degrades performance of processes. We should drop index and disable constraints before start of long running process. We should recreate index and enable constraints after the completion of long running process.

2. Minimize number of Informatica lookups. This eliminates opportunity of SQL optimization at database level.

3. Database partitioning techniques helps improve system availability at application level itself.

4. Technical design should be driven by functional requirement and should be futuristic.

**About Mootly Inc.**

Mootly heavily invests in research and engage specialists to deliver highly usable technology solutions and exceptional services that create value for our customer’s business.

Our focus is product development and IT consulting in area of WCM, Data Migration & Services, and Business Analytics. We partner with customers, technology specialists, college students, and software & hardware vendors to create solutions that solve the customer’s business problems.


For more information, please visit www.mootly.com or call +1-408-475-2035.

**Disclaimer:** Author accepts no liability for the content of this article, or for the consequences of any actions taken on the basis of the information provided, unless that information is subsequently confirmed in writing. Any views or opinions presented in this article are solely those of the author and do not necessarily represent those of the company or any other organization.

**About Author**

This case study is written by Atul Srivastav, Technical Architect, who evaluated existing technical architecture & system and made changes to it to achieve desired result.

Please contact author at +1-214-886-3932 for any question on this case study and for references.