Mapping Optimization By Logical Data Transformation Manager
Abstract

This document describes the methods that the Logical Data Transformation manager (LDTM) uses to optimize mappings. It also explains how optimization affects mapping performance.

Supported Versions

- Data Services 10.0
- Data Quality 10.0
- Big Data Management 10.0

Table of Contents

Overview .......................................................................................... 2
Optimization Methods ........................................................................ 2
Optimizer Levels ................................................................................. 3
Global Predicate Optimization .......................................................... 3
Predicate Optimization ...................................................................... 4
Early Selection Optimization .............................................................. 5
Early Projection Optimization ............................................................ 5
Cost-Based Optimization ................................................................... 5
Push-Into Optimization ...................................................................... 5
Branch-Pruning Optimization ............................................................ 5
Semi-Join Optimization .................................................................... 6
Dataship-Join Optimization .............................................................. 6
Transformation Side Effects ............................................................... 7

Overview

The Data Integration Service uses the Logical Data Transformation Manager (LDTM) component to optimize mappings. When the LDTM optimizes a mapping, it reduces the number of rows that the Data Transformation Manager (DTM) must process.

A mapping is a set of inputs and outputs that represent the data flow between sources and targets. They can be linked by transformation objects that define the rules for data transformation. The Data Integration Service uses the instructions configured in the mapping to read, transform, and write data. There are different kinds of mappings:

- Logical mapping. A logical mapping is the mapping that the LDTM executes. It represents a Data Integration task. It contains constructs such as complex types and logical transformations.
- Executable mapping. An executable mapping is the mapping that the EDTM executes. It contains only flat types and all its transformations are directly understood by EDTM.

Optimization Methods

The LDTM applies optimization methods in the following order:

1. Extract source constraints
2. Simplify expressions
3. Early uncorrelated subquery
4. Global predicate
   a. Predicate inference and filter
   b. Early selection
   c. Predicate minimization
5. Simplify expressions
6. Push-into
7. Early projection
8. Cost-based
9. Branch pruning
10. Pushdown
11. Semi-join and data-ship join

**Optimizer Levels**

The LDTM applies different optimizer methods based on the optimizer level that you configure for a mapping. The following table describes the optimization methods that are applied at the different optimizer levels:

<table>
<thead>
<tr>
<th>Optimization Method</th>
<th>None</th>
<th>Minimal</th>
<th>Normal (default)</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Predicate</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Predicate</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Early Selection</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Early Projection</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Cost-Based</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>Push-Into</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Branch Pruning</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Semi-Join</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Dataship-Join</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Global Predicate Optimization**

Global predicate optimization removes rows that can be filtered out as early as possible in the mapping. This reduces the number of rows that the DTM must process. The global predicate optimization method includes the predicate optimization and early selection methods.
For example, a mapping contains a Joiner transformation with the join condition "A=B" and a Filter transformation with the filter condition "A>5." The LDTM can add "B>5" to the join condition and move the Filter transformation closer to the source.

Global predicate optimization applies predicate expressions more effectively than predicate optimization. Global predicate optimization simplifies or rewrites expressions to increase mapping performance. It also applies predicate expressions as early as possible in the mapping to improve mapping performance.

The global predicate optimization method infers filters and pushes them closer to the source when the mapping contains nested joiners or branches with filters on each branch. When the LDTM applies global predicate optimization, it splits the filters, moves the filters closer to the source, or removes the filters from the mapping.

If the transformation has side effects, or if the LDTM cannot determine whether the transformation has side effects, then the LDTM does not apply global predicate optimization. To apply global predicate optimization to the transformation, you must manually enable optimization.

**Global Predicate Optimization**

**Predicate Optimization**

Predicate optimization simplifies or rewrites predicate expressions to increase mapping performance. To apply predicate optimization, the LDTM infers relationships from existing predicate expressions and creates new predicate expressions.

For example, a mapping contains a Joiner transformation with the join condition "A=B" and a Filter transformation with the filter condition "A>5." The LDM can add "B>5" to the join condition.

The LDTM applies predicate optimization with early selection optimization when it can apply both methods to a mapping.

For example, when predicate optimization creates new filter conditions, the LDTM uses early selection optimization to move the filter conditions closer to the source.
Early Selection Optimization

Early selection is a filter optimization that moves filters as close to the source as possible in a mapping. To do this, the LDTM might split, move, or remove Filter transformations from the mapping.

For example, the LDTM might split a Filter transformation if the filter condition is a conjunction. For example, the filter condition "A>100 AND B<50" might be split into the conditions "A>100" and "B<50." When the LDTM splits a filter, it moves the simplified filters up the mapping pipeline, closer to the source. The LDTM moves the filters up the pipeline separately when it splits the filter.

Early selection optimization can be affected by mappings that have side effects. If the filter occurs before the side effect function, then the mapping might have unexpected results.

Early Projection Optimization

Early projection optimization removes links between ports that do not supply data to the target. Early projection optimization improves performance by reducing the amount of data that the Data Integration Service moves across transformations.

Cost-Based Optimization

Cost-based optimization evaluates a mapping, generates semantically equivalent mappings, and runs the mapping that has the best performance. To determine which mapping has the best performance, the LDTM uses profiling or database statistics to compute the cost for the original mapping and alternate mappings. The LDTM validates the best alternate mapping to ensure that it produces the same results as the original mapping.

The LDTM caches the best alternate mapping in memory. When you run a mapping, the LDTM retrieves the alternate mapping and runs it instead of the original mapping.

Cost-based optimization reduces run time for mappings that have inner-joins and full outer joins. Cost-based optimization can also apply a sorted merge join instead of a nested loop join if it determines that the sorted merge join performs better than the nested loop join.

Push-Into Optimization

Push-into optimization is a filter optimization method that moves Filter transformation logic into the transformation immediately upstream of the Filter transformation in the mapping.

The LDTM does not move filter logic into another transformation if the transformation has side effects. If the filter occurs before the side effect function, then the mapping might have unexpected results.

Branch-Pruning Optimization

The LDTM applies branch pruning optimization to transformations that do not contribute rows to the mapping target.

The LDTM might remove a Filter transformation if the filter condition evaluates to FALSE for the data rows. For example, a mapping has two Filter transformations that filter data from two relational sources. A Filter transformation has the filter condition Country=US, and the other Filter transformation has the filter condition Country=Canada. A Union transformation joins the two relational sources and has the filter condition Country=US.

The Data Integration Service might remove the Filter transformation with the filter condition Country=Canada from the mapping.

By default, branch pruning optimization is enabled when you configure the Developer tool to use the normal or full optimizer level. To disable branch pruning, set the optimizer level at minimal or none.
If the transformation has side effects, or if the LDTM cannot determine whether the transformation has side effects, then the LDTM does not apply branch pruning optimization. To apply branch pruning optimization to the transformation, you must manually enable optimization.

**Union Transformation**

Initial Mapping:

```
Query: select * from orders_op where o_orderstatus='P'
```

Optimized Mapping:

```
Note: Flat file sources
```

**Semi-Join Optimization**

Semi-join optimization reduces the amount of data extracted from the mapping source by modifying join operations in the mapping.

The LDTM applies the semi-join optimization method to a Joiner transformation when one input group has many more rows than the other and when the larger group has many rows with no match in the smaller group based on the join condition. The LDTM decreases the size of the data set of one join operand by reading the rows from the smaller group, finding the matching rows in the larger group, and then performing the join operation. Decreasing the size of the data set improves mapping performance because the LDTM no longer reads unnecessary rows from the larger group source. The LDTM moves the join condition to the larger group source and reads only the rows that match the smaller group.

Before applying the semi-join optimization method, the LDTM determines whether semi-join optimization is possible and likely to be worthwhile. If the analyses determine that this method is likely to improve performance, then the LDTM applies it to the mapping. The LDTM then reanalyzes the mapping to determine whether there are additional opportunities for semi-join optimization. It performs additional optimizations if appropriate.

By default, semi-join optimization is disabled.

**Dataship-Join Optimization**

The LDTM applies dataship-join optimization when there is a significant size difference between two tables.

For example, the LDTM can apply the dataship-join optimization method to join a master table that contains 10,000 rows with a detail table that contains 1,000,000 rows. To perform the dataship-join, the LDTM creates a temporary staging table in the database that contains the larger detail table. Then, the LDTM copies the smaller master table to a
temporary table and joins the data in the temporary table with the data in the larger detail table. After the LDTM performs the join operation, the Joiner transformation logic is processed in the database.

The dataship-join optimization method does not always increase performance. The following factors affect mapping performance with dataship-join optimization:

- The Joiner transformation master source must have significantly fewer rows than the detail source.
- The detail source must be significantly large to justify the optimization. If the detail source is not large enough, then it is faster for the LDTM to read all the data from the master and detail source without applying the dataship-join optimization method.

The LDTM applies dataship-join optimization to a Joiner transformation if the transformation meets the following requirements:

- The join type is normal, master outer, or detail outer.
- The detail pipeline originates from a relational source.
- If the mapping uses target-based commits, the Joiner transformation scope is All Input.
- The master and detail pipelines do not share a transformation.
- The mapping does not contain a branch between the detail source and the Joiner transformation.

**Transformation Side Effects**

Transformation side effects can limit when the LDTM can optimize the mapping. A transformation has side effects if it returns rows and modifies an object, or if it interacts with other objects or functions. For example, the transformation might modify a database, add to a total, raise an exception, write an email, or call other functions with side effects.

If the LDTM applies certain types of optimization to a mapping that has side effects, then it can alter the mapping results. The following optimization types can be affected by mappings that have side effects: early selection, branch pruning, global predicate optimization, and push-into optimization. In early selection and push-into optimization, the LDTM moves filter logic from a Filter transformation as close to the source as possible. If the filter occurs before the side effect function, then the mapping results change.

Before LDTM optimizes a mapping, the LDTM identifies which transformations have side effects. If the LDTM cannot determine if the transformation has side effects, then the LDTM assumes that the transformation has side effects. The LDTM cannot determine if SQL transformations, Web Service Consumer transformations, and Java transformations have side effects. You must manually configure optimization for these transformations.

**Authors**

Thimma Reddy Kalva  
Performance Engineer

Melufa M.M  
Documentation Trainee