Informatica®
10.2.1

Performance Tuning Guide

Informatica Performance Tuning Guide
10.2.1
May 2018

Publication Date: 2019-04-29

Table of Contents

Preface

The *Informatica Performance Tuning Guide* is written for administrators and developers who are interested in improving performance. This guide assumes you have knowledge of the operating systems, networks, client tools, relational database concepts, and flat files in your environment. For more information about database performance tuning not covered in this guide, see the documentation accompanying your database products.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Performance Tuning Overview

This chapter includes the following topics:

Performance Tuning Overview

The goal of performance tuning is to eliminate performance bottlenecks. A bottleneck is an area in the mapping that runs the most frequently and has the lowest throughput. A bottleneck reduces the overall performance of the mapping.

To optimize a mapping, identify a performance bottleneck, eliminate it, and then identify the next performance bottleneck. Optimize one mapping component at a time. You can time a mapping before and after the change to verify that the optimization has a performance impact.

You can find and fix bottlenecks in mapping components such as sources, targets, and connections. You can check bottlenecks in the Data Integration Service and the machine that the Data Integration Service runs on. You can also tune properties for web services and SQL data services.

Performance Tuning Process

You can follow a set of steps to tune mapping components and increase performance.

You might optimize mapping components in the following order:

1. Targets

2. Sources
3. Mappings
4. Transformations
5. Informatica environment in the Administrator tool
6. The computer system
7. Data Service or web service

Use the following methods to identify performance bottlenecks:

- Run test mappings. You can configure a test mapping to read from a flat file source or to write to a flat file target to identify source and target bottlenecks.
- Analyze performance details. Analyze performance details, such as optimization methods, to determine where mapping performance decreases.
- Monitor system performance. You can use system monitoring tools to view the percentage of CPU use, I/O waits, paging, and system resource usage.

Target Bottlenecks

Target bottlenecks are decreases in performance when the Data Integration Service writes to a target. Target bottlenecks might occur when the database uses small checkpoint intervals or small database network packet sizes.

The most common performance bottleneck occurs when the Data Integration Service writes to a target database. If the database uses small checkpoint intervals, the database processing slows more often write a checkpoint. Small database network packet sizes can cause bottlenecks. You can allow larger packets of data to cross the network at one time.

To identify a target bottleneck, you can create a copy of the mapping that has a flat file target instead of a database target. If thes performance increases significantly, you have a target bottleneck. If the mapping already writes to a flat file target, you probably do not have a target bottleneck.

Source Bottlenecks

Source bottlenecks are performance decreases when the Data Integration Service reads from a source database. Source bottlenecks might occur when the source query is not efficient or when the database network packet sizes are small.

When the mapping reads from a relational source, you can use the following methods to identify source bottlenecks:

- Add a Filter transformation to the mapping. Add the Filter transofrmation after the source. Set the Filter condition to false so the Filter transformation does not return any data. If the amount of time that the mapping takes is about the same, the mapping has a source bottleneck.
- Create a read test mapping. Make a copy of the mapping but remove all the transformations, joins, or queries. Connect the source to a target. If the mapping performance is similar to the original mapping, you have a source bottleneck.

- Run the read query directly against the source database. Copy the read query from the mapping log. Run the query against the source database with a query tool such as isql. Measure the run time and the time it takes for the query to return a row.

Mapping Bottlenecks

If you determine that you do not have a source or target bottleneck, you may have a mapping bottleneck. Small cache size, low buffer memory, and small commit intervals can cause mapping bottlenecks.

To identify a mapping bottleneck, analyze the performance details in the mapping log. Performance details include information about each transformation, such as the number of input rows, output rows, and error rows.

You can also add a Filter transformation before each target definition. Set the filter condition to false so the Filter transformation loads no data into the target tables. If the time it takes to run the new mapping is the same as the original mapping, you have a mapping bottleneck.

Computer System Bottlenecks

You can view resource usage when you run Informatica services on Windows or UNIX . On Windows use the Task Manager. UNIX has multiple tools that you can use to review performance.

Identifying System Bottlenecks on Windows

You can view the Performance and Processes tab in the Task Manager for system information. The Performance tab in the Task Manager provides an overview of CPU usage and total memory used. Use the Performance Monitor to view more detailed information.

The following table describes the system information that you can use in the Windows Performance Monitor to create a chart:

Property	Description
Percent processor time	If you have more than one CPU, monitor each CPU for percent processor time.
Pages/second	If pages/second is greater than five, you may have excessive memory pressure known as thrashing.
Physical disks percent time	The percent of time that the physical disk is busy performing read or write requests.
Physical disks queue length	The number of users waiting for access to the same disk device.
Server total bytes per second	The server has sent to and received from the network.

Identifying System Bottlenecks on UNIX

Use the following tools to identify system bottlenecks on UNIX:

- top. View overall system performance. This tool displays CPU usage, memory usage, and swap usage for the system and for individual processes running on the system.
- iostat. Monitor the loading operation for every disk attached to the database server. Iostat displays the percentage of time that the disk is physically active. If you use disk arrays, use utilities provided with the disk arrays instead of iostat.
- vmstat. Monitor disk swapping actions.
- sar. View detailed system activity reports of CPU, memory, and disk usage. You can use this tool to monitor CPU loading. It provides percent usage on user, system, idle time, and waiting time. You can also use this tool to monitor disk swapping actions.

Run-time Bottlenecks

Enable performance features and tune Data Integration Service properties to optimize mapping performance. Configure optimization settings for the Data Integration Service and the Model Repository Service in the Administrator tool.

Allocate memory for optimal system performance and configure error tracing levels to reduce the number of log events generated by the Data Integration Service when it runs the mapping.

You can configure the maximum amount of memory that the Data Integration Service allocates for running all the concurrent requests. You can also limit the maximum amount of memory that the Data Integration Service allocates for any given request.

You can configure the result set cache, to enable the Data Integration Service to cache the results of the DTM process associated with each SQL data service query and web service request.

SQL Data Service Optimization Bottlenecks

You can optimize SQL data services to improve performance when end users run SQL queries against them using third-party client tools. If an SQL data service uses a virtual table mapping, you can optimize the transformations and the mapping.

You can optimize the JDBC driver to improve performance when querying an SQL data service. You can also configure the data object cache for the Data Integration Service to improve the performance of mappings and SQL queries.

Web Service Optimization Bottlenecks

You can optimize web services to improve performance when the Data Integration Service runs web service requests. Tune the Data Integration Service to manage memory, handle concurrent web service requests, and keep a DTM process active so that it can process more than one web service request.

To improve web service performance, use web service message compression, optimize HTTP requests, and configure the data object cache.

Connection Bottlenecks

You can optimize connections to improve performance. You can manage the pool of idle connection instances for a database connection. You can increase the network packet size to allow larger packets of data to cross the network at one time.

CHAPTER 2

Target Optimization

This chapter includes the following topics:

Target Optimization Overview

Optimize targets to enable the Data Integration Service to write to the targets efficiently. You can drop indexes and key constraints before running a mapping, increase the number of checkpoint intervals in the database, configure bulk loading in the write properties for a data object, and optimize an Oracle target database.

Use the following optimization techniques to optimize the target:

- Optimize flat file targets.
- Increase database checkpoint intervals.
- Use bulk loads.
- Optimize Oracle target databases.

Flat File Target Optimization

You can improve mapping performance by optimizing flat file targets. You can also push transformation tasks to a command to improve performance.

Consider the following solutions to reduce flat file target bottlenecks:

Push transformation tasks to a command instead of the Data Integration Service.

You can improve mapping performance by pushing transformation tasks to a command instead of the Data Integration Service. You can also use a command to sort or to compress target data. In the Developer tool, configure the Command property in the run-time properties for a flat file target.

On UNIX, use any valid UNIX command or shell script. On Windows, use any valid DOS command or batch file. The flat file writer sends the data to the command instead of a flat file target.

For example, use the following command to generate a compressed file from the target data:

```
compress -c - > MyTargetFiles/MyCompressedFile.Z
```

Write to a flat file target that is local to the service process node.

If the Data Integration Service runs on a single node and writes to a flat file target, you can optimize mapping performance by writing to a flat file target that is local to the service process node.

Database Checkpoint Intervals

The Data Integration Service performance slows each time it waits for the database to perform a checkpoint.

Consider the following solution to reduce database checkpoint bottlenecks:
Increase the checkpoint interval in the database.

To decrease the number of checkpoints and increase performance, increase the checkpoint interval in the database.

Although you gain performance when you reduce the number of checkpoints, you also increase the recovery time if the database shuts down unexpectedly.

Bulk Loads

When you use bulk loading, the Data Integration Service bypasses the database log, which speeds performance.

Consider the following solutions to reduce bulk load bottlenecks:
Configure bulk loading in the write properties for a data object.

You can use bulk loading to improve the performance of a mapping that inserts a large amount of data into a DB2, Sybase ASE, Oracle, or Microsoft SQL Server database.

Without writing to the database log, the target database cannot perform rollback. As a result, you may not be able to perform recovery. When you use bulk loading, weigh the importance of improved mapping performance against the ability to recover an incomplete mapping.

Database Target Optimization

You can optimize the target database by checking the storage clause, space allocation, and rollback or undo segments.

Consider the following solutions to reduce database target bottlenecks:
Verify that the database stores rollback or undo segments in appropriate tablespaces, preferably on different disks.

When you write to the database, the database uses rollback or undo segments during loads. Ask the database administrator to ensure that the database stores rollback or undo segments in appropriate tablespaces, preferably on different disks. The rollback or undo segments should also have appropriate storage clauses.

Tune the database redo log.

To optimize the database, tune the database redo log. The database uses the redo log to log loading operations. Make sure the redo log size and buffer size are optimal. For an Oracle database, you can view redo log properties in the init.ora file.

Connect to an Oracle database with the IPC protocol.

If the Data Integration Service runs on a single node and the Oracle instance is local to the service process node, you can optimize performance by using IPC protocol to connect to the Oracle database. You can set up Oracle database connection in listener.ora and tnsnames.ora.

CHAPTER 3

Source Optimization

This chapter includes the following topics:

Source Optimization Overview

Optimize flat file, relational and custom data sources to enable the Data Integration Service to read source data efficiently.

Use the following optimization techniques to optimize sources:

- Read source data efficiently.
- Use query optimization techniques.
- Use conditional filters with the SQL query.
- Select unique values from the source.
- Apply hints to the SQL query.
- Configure constraints on logical data objects, physical data objects, and virtual tables.
- Configure customized data objects for optimization.
- Configure Oracle, Sybase, and Microsoft SQL Server databases for optimization.

Flat File Source Optimization

Configure the format properties for flat file sources to enable the Data Integration Service to read source data efficiently.

Consider the following solutions for flat file source bottlenecks:

Do not use quotes or escape characters in the format properties for a delimited flat file.

If you specify an escape character, the Data Integration Service reads the delimiter character as a regular character embedded in the string. You can improve mapping performance slightly if the source file does not contain quotes or escape characters.

Set the number of bytes the Data Integration Service reads per line.

If the mapping reads from a flat file source, you can improve mapping performance by setting the number of bytes the Data Integration Service reads per line. Configure the Line Sequential Buffer Length property in the run-time properties for flat file sources.

By default, the Data Integration Service reads 1024 bytes per line. If each line in the source file is less than the default setting, you can decrease the line sequential buffer length in the mapping properties.

Query Optimization

If a mapping joins multiple source tables in one customized data object, you might be able to improve performance by optimizing the query with optimizing hints. Also, single table select statements with an ORDER BY or GROUP BY clause may benefit from optimization such as adding indexes.

Consider the following solutions for query bottlenecks:

Create optimizer hints to tell the database how to execute the query for a particular set of source tables.

Usually, the database optimizer determines the most efficient way to process the source data. However, you might know properties about the source tables that the database optimizer does not. The database administrator can create optimizer hints to tell the database how to execute the query for a particular set of source tables.

Configure optimizer hints to begin returning rows as quickly as possible, rather than returning all rows at once.

Use optimizing hints if there is a long delay between when the query begins executing and when the Data Integration Service receives the first row of data. Configure optimizer hints to begin returning rows as quickly as possible, rather than returning all rows at once. This allows the Data Integration Service to process rows parallel with the query execution.

Create an index on the ORDER BY or GROUP BY columns.

Queries that contain ORDER BY or GROUP BY clauses may benefit from creating an index on the ORDER BY or GROUP BY columns. Once you optimize the query, use the SQL override option to take full advantage of these modifications.

Configure the database to run parallel queries.

You can also configure the source database to run parallel queries to improve performance. For more information about configuring parallel queries, see the database documentation.

The query that the Data Integration Service uses to read data appears in the virtual database in a SQL Data Service. You can also find the query in the customized data object. Have the database administrator analyze the query, and then create optimizer hints and indexes for the source tables.

Conditional Filters

A simple source filter on the source database can sometimes negatively impact performance because of the lack of indexes. You can use the conditional filter in the customized data object to improve performance.

Consider the following solution for conditional filter bottlenecks:

Use the conditional filter for multiple mappings that read from the same source simultaneously.

If multiple mappings read from the same source simultaneously, the conditional filter may improve performance.

However, some mappings may perform faster if you filter the source data on the source database. You can test the mapping with both the database filter and the conditional filter to determine which method improves performance.

Select Distinct

You can select unique values from sources in a customized data object through the select distinct option. When you use select distinct, the Data Integration Service adds a SELECT DISTINCT statement to the default SQL query.

Consider the following solution for Select Distinct bottlenecks:

Use the Select Distinct option to filter unnecessary data earlier in the data flow.

Use the Select Distinct option for the customized data object if you want the Data Integration Service to select unique values from a source. Use the Select Distinct option to filter unnecessary data earlier in the data flow. This can improve performance.

For example, you might use the select distinct option to extract unique customer IDs from a table that lists total sales. When you use the customized data object in a mapping, the Data Integration Service filters out unnecessary data earlier in the data flow, which can increase performance.

Hints

You can add hints to the source SQL query to pass instructions to a database optimizer. The optimizer uses the hints to choose a query run plan to access the source.

The Hints field appears in the **Query** view of a relational data object instance or a customized data object. The source database must be Oracle, Sybase, IBM DB2, or Microsoft SQL Server. The Hints field does not appear for other database types.

When the Data Integration Service generates the source query, it adds the SQL hints to the query exactly as you enter them in the Developer tool. The Data Integration Service does not parse the hints. When you run the mapping that contains the source, the mapping log shows the query with the hints in the query.

The Data Integration Service inserts the SQL hints in a position in the query depending on the database type. Refer to your database documentation for information about the syntax for hints.

Oracle

The Data Integration Service add hints directly after the SELECT/UPDATE/INSERT/DELETE keyword.

```
SELECT /*+ <hints> */ FROM …
```

The '+' indicates the start of hints.

The hints are contained in a comment (/* ... */ or --... until end of line)

Sybase

The Data Integration Service adds hints after the query. Configure a plan name in the hint.

```
SELECT … PLAN <plan>
```

```
select avg(price) from titles plan "(scalar_agg (i_scan type_price_ix titles )"
```

IBM DB2

You can enter the optimize-for clause as a hint. The Data Integration Service adds the clause at the end of the query.

```
SELECT … OPTIMIZE FOR <n> ROWS
```

The optimize-for clause tells the database optimizer how many rows the query might process. The clause does not limit the number of rows. If the database processes more than <n> rows, then performance might decrease.

Microsoft SQL Server

The Data Integration Service adds hints at the end of the query as part of an OPTION clause.

```
SELECT … OPTION ( <query_hints> )
```

Hints Rules and Guidelines

Use the following rules and guidelines when you configure hints for SQL queries:

- If you enable pushdown optimization or if you use a semi-join in a relational data object, then the original source query changes. The Data Integration Service does not apply hints to the modified query.
- You can combine hints with join and filter overrides, but if you configure a SQL override, the SQL override takes precedence and the Data Integration Service does not apply the other overrides.
- The **Query** view shows a simple view or an advanced view. If you enter a hint with a filter, sort, or join override on the simple view, and you the Developer tool shows the full query override on the advanced view.

Creating Hints

Create hints to send instructions to the database optimizer to determine a query plan.

1. Open the customized data object or the relational data object instance.
2. Select the **Read** view.
3. Select the Output transformation.
4. Select the **Query** properties.
5. Select the simple query.
6. Click **Edit** next to the **Hints** field.

   The **Hints** dialog box appears.
7. Enter the hint in the **SQL Query** field.

   The Developer tool does not validate the hint.

8. Click **OK**.
9. Save the data object.

Constraints

The Data Integration Service can read constraints from relational sources, flat file sources, logical data objects, or virtual tables. A constraint is a conditional expression that the values on a data row must satisfy.

When the Data Integration Service reads constraints, it might drop the rows that do not evaluate to TRUE for the data rows based on the optimization method applied.

Before you set a constraint, you must verify that the source data satisfies the condition set by the constraint.

For example, a source database has an AGE column that appears to have rows with AGE < 70. You can set a constraint with AGE < 70 on the source database. The Data Integration reads records from the source database with the constraint AGE < 70. If the Data Integration Service reads records with AGE >= 70, it might drop the rows with AGE >= 70.

In the database, you can use SQL commands to set constraints on the database environment when you connect to the database. The Data Integration Service runs the connection environment SQL each time it connects to the database.

Use the Developer tool to set constraints on logical data objects, physical data objects, and virtual tables. When you set a constraint, you must enter an expression that evaluates to TRUE for each data row.

Configuring Constraints

You can add constraints to relational data objects, flat file data objects, customized data objects, logical data objects, and virtual tables. After you add a constraint, you can edit or delete the constraint.

1. From the **Object Explorer** view, open the mapping that contains the relational data object added as a Read transformation. Or, open the flat file data object, customized data object, logical data object, or virtual table.
   - To set constraints on a relational data object added to a mapping as a Read transformation, select the Read transformation in the mapping. On the **Properties** view, select the **Advanced** tab.
   - To set constraints on a flat file data object, select the **Advanced** view and expand the **Run-time: Read** section.
   - To set constraints on a customized data object, select the **Read** view and select the **Output** port of the source transformation. On the **Properties** view, select the **Advanced** tab.
   - To set constraints on a logical data object, select a logical data model and select the logical data object. On the **Properties** view, select the **Advanced** tab.
   - To set constraints on a virtual table, open the virtual table from the SQL endpoint. On the **Properties** view, select the **Advanced** tab.
2. Click the value field for constraints.

   The **Constraints** dialog box appears.
3. Click **New** to open the Expression editor.
4. Configure the constraint logic and use expression functions and columns as parameters.
5. Click **Validate**.

6. Click **OK**.

Customized Data Object Optimization

You can configure customized data objects to improve performance. You can optimize the SQL query, use conditional filters, and select distinct values from the source in a customized data object.

Consider the following solutions for customized data object bottlenecks:

Create a custom query to issue a special SELECT statement for the Data Integration Service to read source data.

The custom query replaces the default query that the Data Integration Service uses to read data from sources.

Filter rows when the Data Integration Service reads source data.

If you include a filter condition, the Data Integration Service adds a WHERE clause to the default query.

Select distinct values from the source.

If you choose Select Distinct, the Data Integration Service adds a SELECT DISTINCT statement to the default SQL query.

Apply database hints.

You can add hints to the source SQL query to pass instructions to a database optimizer.

Configure constraints on source data.

If you configure constraints on flat files and relational tables in a customized data object, the Data Integration Service drops the rows that do not evaluate to TRUE for the data rows.

Database Source Optimization

If the source database is Oracle, you can optimize the Data Integration Service performance, by using the IPC protocol to connect to the Oracle database. You can also move the temporary database to a disk array to improve performance.

Consider the following solutions for database source bottlenecks:

Use IPC protocol to connect to the Oracle database.

If the Data Integration Service runs on a single node and the Oracle instance is local to the service process node, you can optimize performance by using IPC protocol to connect to the Oracle database. You can set up Oracle database connection in listener.ora and tnsnames.ora.

Move the temporary database and redo logs to a disk array or faster drives.

When you join large tables on a database, you can use a redundant array of independent disks (RAID) for the cache location. Alternatively, you can add more files to the primary file group on other disks to divide the load between the disks.

CHAPTER 4

Transformation Optimization

This chapter includes the following topics:

Transformation Optimization

Optimize transformations to enable the Data Integration Service to process transformations in a mapping efficiently.

Use the following optimization techniques to optimize the transformation:

- Configure transformations for optimization.
- Eliminate transformation errors.
- Configure the transformation cache.

Aggregator Transformation Optimization

Aggregator transformations often slow performance because they must group data before processing it. Aggregator transformations need additional memory to hold intermediate group results.

Consider the following solutions for Aggregator transformation bottlenecks:

Group by simple columns.

You can optimize Aggregator transformations when you group by simple columns. When possible, use numbers instead of string and dates in the columns used for the GROUP BY. Avoid complex expressions in the Aggregator expressions.

Use sorted input.

To increase mapping performance, sort data for the Aggregator transformation. Use the Sorted Input option to sort data.

The Sorted Input option decreases the use of aggregate caches. When you use the Sorted Input option, the Data Integration Service assumes all data is sorted by group. As the Data Integration Service reads rows for a group, it performs aggregate calculations. When necessary, it stores group information in memory.

The Sorted Input option reduces the amount of data cached during the mapping and improves performance. Use the Sorted Input option or a Sorter transformation to pass sorted data to the Aggregator transformation.

You can increase performance when you use the Sorted Input option in mappings with multiple partitions.

Filter data before you aggregate it.

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

Limit port connections.

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

Expression Optimization

Some expressions used in a transformation might decrease performance.

Consider the following solutions for expression bottlenecks:

Isolate slow expressions.

Slow expressions slow mapping performance. To isolate slow expressions, remove expressions from the mapping one at a time, and run the mapping to determine the time it takes to run the mapping without the expression. If there is a significant difference in mapping run time, look for ways to optimize the slow expression.

Complete the following steps to evaluate expression performance:

1. Time the mapping with the original expressions.
2. Copy the mapping and replace half of the complex expressions with a constant.
3. Run and time the edited mapping.
4. Make another copy of the mapping and replace the other half of the complex expressions with a constant.
5. Run and time the edited mapping.

Factor out common logic.

If the mapping performs the same task in multiple places, reduce the number of times the mapping performs the task by moving the task earlier in the mapping. For example, you have a mapping with five

target tables. Each target requires a Social Security number lookup. Instead of performing the lookup five times, place the Lookup transformation in the mapping before the data flow splits. Next, pass the lookup results to all five targets.

Minimize aggregate function calls.

When writing expressions, factor out as many aggregate function calls as possible. Each time you use an aggregate function call, the Data Integration Service must search and group the data. For example, in the following expression, the Data Integration Service reads COLUMN_A, finds the sum, then reads COLUMN_B, finds the sum, and finally finds the sum of the two sums:

```
SUM(COLUMN_A) + SUM(COLUMN_B)
```

If you factor out the aggregate function call, as below, the Data Integration Service adds COLUMN_A to COLUMN_B, then finds the sum of both.

```
SUM(COLUMN_A + COLUMN_B)
```

Replace common expressions with local variables.

If you use the same expression multiple times in one transformation, you can make that expression a local variable. You can use a local variable only within the transformation. However, by calculating the variable only once, you speed performance.

Choose numeric versus string operators.

The Data Integration Service processes numeric operations faster than string operations. For example, if you look up large amounts of data on two columns, EMPLOYEE_NAME and EMPLOYEE_ID, configuring the lookup around EMPLOYEE_ID improves performance.

Optimize CHAR-CHAR and CHAR-VARCHAR comparisons.

When the Data Integration Service performs comparisons between CHAR and VARCHAR columns, it slows each time it finds trailing blank spaces in the row. You can use the TreatCHARasCHARonRead option when you configure the Data Integration Service in the Informatica Administrator so that the Data Integration Service does not trim trailing spaces from the end of Char source fields.

Choose DECODE versus LOOKUP.

When you use a LOOKUP function, the Data Integration Service must look up a table in a database. When you use a DECODE function, you incorporate the lookup values into the expression so the Data Integration Service does not have to look up a separate table. Therefore, when you want to look up a small set of unchanging values, use DECODE to improve performance.

Use operators instead of functions.

The Data Integration Service reads expressions written with operators faster than expressions with functions. Where possible, use operators to write expressions. For example, you have the following expression that contains nested CONCAT functions:

```
CONCAT( CONCAT( CUSTOMERS.FIRST_NAME, ' ') CUSTOMERS.LAST_NAME)
```

You can rewrite that expression with the || operator as follows:

```
CUSTOMERS.FIRST_NAME || ' ' || CUSTOMERS.LAST_NAME
```

Optimize IIF functions.

IIF functions can return a value and an action, which allows for more compact expressions. For example, you have a source with three Y/N flags: FLG_A, FLG_B, FLG_C. You want to return values based on the values of each flag.

You use the following expression:

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y',
VAL_A + VAL_B + VAL_C,
```

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N',
VAL_A + VAL_B ,
IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y',
VAL_A + VAL_C,
IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N',
VAL_A ,
IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y',
VAL_B + VAL_C,
IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N',
VAL_B ,
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y',
VAL_C,
IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N',
0.0,
))))))))
```

Java Transformation Optimization

Some Java transformations in a mapping might decrease performance.

Consider the following solution to increase Java transformation performance:
Enable early selection or push-into filter optimization methods with the Java transformation.

You can enable early selection or push-into optimization in Java transformations. Update the code snippets on the **Optimizer Interfaces** tab of the Java transformation.

Early Selection Optimization with the Java Transformation

You can enable an active or passive Java transformation for early selection optimization if the Java transformation has no side effects. The optimizer passes the filter logic through the Java transformation and modifies the filter condition as required.

To view the code snippets for early selection optimization, choose PredicatePushOptimization in the navigator of the **Optimizer Interfaces** tab.

allowPredicatePush

Boolean. Enables early selection. Change the function to return a true result and message in order to enable early selection. Default is false, and the function returns a message that optimization is not supported.

```
public ResultAndMessage allowPredicatePush(boolean ignoreOrderOfOp) {
    // To Enable PredicatePushOptimization, this function should return true
    //return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Predicate Push Optimization Is Not
Supported");
}
```

canGenerateOutputFieldEvalError

Boolean. Indicates whether or not the Java transformation can return an output field error, such as a division by zero error. Change the function to return false if the Java transformation does not generate output field errors. When the Java transformation can generate field errors, then the Data Integration Service cannot use early selection optimization.

```
public boolean canGenerateOutputFieldEvalError() {
    // If this Java transformation can never generate an output field evaluation error,
    // return false.
    return true;
}
```

getInputExpr

Returns an Informatica expression that describes which input values from input fields comprise an output field. The optimizer needs to know which input fields comprise an output field in order to push the filter logic through the transformation.

```
public InfaExpression getInputExpr(TransformationField field,
        TransformationDataInterface group) {
    // This should return an Informatica expression for output fields in terms of input
fields
    // We will only push predicate that use fields for which input expressions are
defined.
    // For example, if you have two input fields in0 and in1 and three output fields
out0, out1, out2
    // out0 is the pass-through of in1, out2 is sum of in1 and in2, and out3 is unknown,
the code should be:
    //if (field.getName().equals("out0"))
    //  return new InfaExpression("in0", instance);
    //else if (field.getName().equals("out1"))
    //    return new InfaExpression("in0 + in1", instance);
    //else if (field.getName().equals("out2"))
    //    return null;
    return null;
}
```

For example, a mapping contains a filter expression, `"out0 > 8"`. Out0 is the value of the out0 output port in the Java transformation. You can define the value of out0 as the value of the in0 input port + 5. The optimizer can push the following expression `"(in0 + 5) > 8"` past the Java transformation with early selection optimization. You can return NULL if an output field does not have input field expression. The optimizer does not push filter expressions past output fields with no input expression.
You might include the following code:

```
if (field.getName().equals("out0"))
     return new InfaExpression("in0 + 5", instance);
else if (field.getName().equals("out2"))
     return null;
```

inputGroupsPushPredicateTo

Returns a list of groups that can receive the filter logic. The Java transformation has one input group. Do not modify this function for the Java transformation.

```
public List<TransformationDataInterface> inputGroupsPushPredicateTo(
        List<TransformationField> fields) {
    // This functions returns a list of input data interfaces to push predicates to.
    // Since JavaTx only has one input data interface, you should not have to modify
this function
    AbstractTransformation tx = instance.getTransformation();
    List<DataInterface> dis = tx.getDataInterfaces();
    List<TransformationDataInterface> inputDIs = new
ArrayList<TransformationDataInterface>();
    for (DataInterface di : dis){
        TransformationDataInterface tdi = (TransformationDataInterface) di;
        if (tdi.isInput())
            inputDIs.add(tdi);
    }
    if(inputDIs.size() == 1)
        return inputDIs;
    else
        return null;
}
```

Push-Into Optimization with the Java Transformation

You can enable an active Java transformation for push-into optimization if it has no side effects and the optimization does not affect the mapping results.

When you configure push-into optimization for the Java transformation, you define a way for the Java transformation to store the filter condition that it receives from the optimizer. Add code that examines the filter condition. If the Java transformation can absorb the filter logic, then the Java transformation passes a true condition back to the optimizer. The optimizer removes the Filter transformation from the optimized mapping.

When you configure the Java transformation you write the code that stores the filter condition as transformation metadata during optimization. You also write the code to retrieve the filter condition at run-time and to drop the rows according to the filter logic.

When you define the Java transformation, you add code for push-into optimization on the Java transformation **Optimizer Interfaces** tab. To access the code snippets for push-into optimization, choose FilterPushdownOptimization in the navigator of the transformation **Optimizer Interfaces** tab.

The Developer tool displays code snippets to enable push-into optimization and to receive the filter condition from the optimizer. Update the code snippets to enable optimization and to save the filter logic as transformation metadata.

isFilterSupported

Returns true to enable push-into optimization. Returns false to disable push-into optimization.
Change the function to return true in order to enable push-into optimization.

```
public ResultAndMessage isFilterSupported() {
    // To enable filter push-into optimization this function should return true
    // return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

pushFilter

Receives the filter condition from the optimizer.
Add code to examine the filter and determine if the filter logic can be used in the transformation. If the transformation can absorb the filter, then use the following method to store the filter condition as transformation metadata:

```
storeMetadata(String key, String data)
```

The key is an identifier for the metadata. You can define any string as a key. The data is the data you want to store in order to determine which rows to drop at run time. For example, the data might be the filter condition that the Java transformation receives from the optimizer.

```
public ResultAndMessage pushFilter(InfaExpression condition) {

    // Add code to absorb the filter
    // If filter is successfully absorbed return new ResultAndMessage(true, ""); and the
optimizer
    // will remove the filter from the mapping
    // If the filter is not absorbed, return new ResultAndMessage(false, msg);
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

Joiner Transformation Optimization

Joiner transformations can slow performance because they need additional space at run time to hold intermediary results.

Consider the following solutions for Joiner transformation bottlenecks:

Designate the master source as the source with fewer duplicate key values.

When the Data Integration Service processes a sorted Joiner transformation, it caches rows for one hundred unique keys at a time. If the master source contains many rows with the same key value, the Data Integration Service must cache more rows, which can decrease performance.

Designate the master source as the source with fewer rows.

The Joiner transformation compares each row of the detail source against the master source. The fewer rows in the master, the fewer iterations of the join comparison occur, which speeds the join process.

Perform joins in a database when possible.

Performing a join in a database is faster than performing a join in during the mapping run. The type of database join that you use can affect performance. Normal joins are faster than outer joins and result in fewer rows. Sometimes, you cannot perform the join in the database, such as joining tables from two different databases or flat file systems.

Join sorted data when possible.

Configure the Joiner transformation to use sorted input. The Data Integration Service increases performance by minimizing disk input and disk output. The greatest performance increase occurs when you work with large data sets. For an unsorted Joiner transformation, designate the source with fewer rows as the master source.

Optimize the join condition.

The Data Integration Service attempts to decrease 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 Data Integration Service no longer reads unnecessary rows from the larger group source. The Data Integration Service moves the join condition to the larger group source and reads only the rows that match the smaller group.

Use the semi-join optimization method.

Use the semi-join optimization method to improve mapping performance 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.

Lookup Transformation Optimization

Lookup transformations can slow performance depending on the lookup cache type and lookup conditions.

Consider the following solutions for Lookup transformation bottlenecks:

Use the optimal database driver.

The Data Integration Service can connect to a lookup table using a native database driver or an ODBC driver. Native database drivers provide better mapping performance than ODBC drivers.

Cache lookup tables for relational or flat file lookups.

To improve lookup performance for relational or flat file sources, enable lookup caching in the transformation. When you enable caching, the Data Integration Service caches the lookup table. When you run the mapping, the Data Integration Service queries the lookup cache instead of the lookup table. When this option is not enabled, the Data Integration Service queries the lookup table on a row-by-row basis.

The result of the lookup query and processing is the same, whether or not you cache the lookup table. However, using a lookup cache can increase mapping performance for smaller lookup tables. In general, you want to cache lookup tables that need less than 300 MB.

Cache lookup tables for logical data object lookups.

To improve lookup performance on a logical data object, you can enable data object caching on the Data Integration Service. When you enable data object caching, the Data Integration Service caches the logical data object. To enable data object caching, you must deploy the mapping to an application, enable caching of the logical data object, and run the mapping with the command `infacmd ms runmapping`. When you run the mapping, the Data Integration Service queries the data object cache instead of the logical data object.

If you run the mapping from the Developer tool, the Lookup transformation queries the logical data object on a row-by-row basis.

Use the appropriate cache type.

Use the following types of caches to increase performance:

- Shared cache. You can share the lookup cache between multiple transformations. You can share an unnamed cache between transformations in the same mapping. You can share a named cache between transformations in the same or different mappings.
- Persistent cache. To save and reuse the cache files, you can configure the transformation to use a persistent cache. Use this feature when you know the lookup table does not change between mapping runs. Using a persistent cache can improve performance because the Data Integration Service builds the memory cache from the cache files instead of from the database.

Enable concurrent caches.

When the Data Integration Service processes mappings that contain Lookup transformations, the Data Integration Service builds a cache in memory when it processes the first row of data in a cached Lookup transformation. If there are multiple Lookup transformations in a mapping, the Data Integration Service creates the caches sequentially when the first row of data is processed by the Lookup transformation. This slows Lookup transformation processing.

You can enable concurrent caches to improve performance. When the number of additional concurrent pipelines is set to one or more, the Data Integration Service builds caches concurrently rather than sequentially. Performance improves greatly when the mappings contain a number of active transformations that may take time to complete, such as Aggregator, Joiner, or Sorter transformations. When you enable multiple concurrent pipelines, the Data Integration Service no longer waits for active mappings to complete before it builds the cache. Other Lookup transformations in the pipeline also build caches concurrently.

Optimize lookup condition matches.

When the Lookup transformation matches lookup cache data with the lookup condition, it sorts and orders the data to determine the first matching value and the last matching value. You can configure the transformation to return any value that matches the lookup condition. When you configure the Lookup transformation to return any matching value, the transformation returns the first value that matches the lookup condition. It does not index all ports as it does when you configure the transformation to return the first matching value or the last matching value.

When you use any matching value, performance can improve because the transformation does not index on all ports, which can slow performance.

Reduce the number of cached rows.

You can reduce the number of rows included in the cache to increase performance. Use the Lookup SQL Override option to add a WHERE clause to the default SQL statement. When you add a WHERE clause to a Lookup transformation that uses a dynamic cache, use a Filter transformation before the Lookup transformation to pass rows into the dynamic cache that match the WHERE clause.

Override the ORDER BY statement.

By default, the Data Integration Service generates an ORDER BY statement for a cached lookup. The ORDER BY statement contains all lookup ports. To increase performance, suppress the default ORDER BY statement and enter an override ORDER BY with fewer columns.

The Data Integration Service always generates an ORDER BY statement, even if you enter one in the override. Place two dashes '--' after the ORDER BY override to suppress the generated ORDER BY statement.

For example, a Lookup transformation uses the following lookup condition:

```
ITEM_ID = IN_ITEM_ID
PRICE <= IN_PRICE
```

The Lookup transformation includes three lookup ports used in the mapping, ITEM_ID, ITEM_NAME, and PRICE. When you enter the ORDER BY statement, enter the columns in the same order as the ports in the lookup condition. You must also enclose all database reserved words in quotes.

Enter the following lookup query in the lookup SQL override:

```
SELECT ITEMS_DIM.ITEM_NAME, ITEMS_DIM.PRICE, ITEMS_DIM.ITEM_ID FROM ITEMS_DIM ORDER
BY
ITEMS_DIM.ITEM_ID, ITEMS_DIM.PRICE --
```

Use a machine with more memory.

To increase mapping performance, run the mapping on a Data Integration Service node with a large amount of memory. Increase the index and data cache sizes as high as you can without straining the machine. If the Data Integration Service node has enough memory, increase the cache so it can hold all data in memory without paging to disk.

Optimize the lookup condition.

If you include more than one lookup condition, place the conditions in the following order to optimize lookup performance:

- Equal to (=)
- Less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=)
- Not equal to (!=)

Filter lookup rows.

To improve performance, create a filter condition to reduce the number of lookup rows retrieved from the source when the lookup cache is built.

Index the lookup table.

The Data Integration Service needs to query, sort, and compare values in the lookup condition columns. The index needs to include every column used in a lookup condition.

You can improve performance for the following types of lookups:

- Cached lookups. To improve performance, index the columns in the lookup ORDER BY statement. The mapping log file contains the ORDER BY statement.
- Uncached lookups. To improve performance, index the columns in the lookup condition. The Data Integration Service issues a SELECT statement for each row that passes into the Lookup transformation.

Optimize multiple lookups.

If a mapping contains multiple lookups, even with caching enabled and enough heap memory, the lookups can slow performance. Tune the Lookup transformations that query the largest amounts of data to improve overall performance.

If the lookup table is on the same database as the source table in your mapping and caching is not feasible, join the tables in the source database rather than using a Lookup transformation.

Sorter Transformation Optimization

Sorter transformations can slow performance when the physical RAM on the Data Integration Service node does not have enough memory allocated to sort data.

Consider the following solution for Sorter transformation bottlenecks:

Allocate sufficient memory.

For optimal performance, configure the Sorter cache size with a value less than or equal to the amount of available physical RAM on the Data Integration Service node. Allocate at least 16 MB of physical memory to sort data using the Sorter transformation. The Sorter cache size is set to 16,777,216 bytes by default. If the Data Integration Service cannot allocate enough memory to sort data, it fails the mapping.

If the amount of incoming data is greater than the amount of Sorter cache size, the Data Integration Service temporarily stores data in the Sorter transformation work directory. The Data Integration Service requires disk space of at least twice the amount of incoming data when storing data in the work directory.

SQL Transformation Optimization

Each time the Data Integration Service processes a new query in a mapping, it calls a function called SQLPrepare to create an SQL procedure and pass it to the database. When the query changes for each input row, it might decrease performance.

Consider the following solutions for SQL transformation bottlenecks:

Do not use transaction statements in an SQL transformation query.

When an SQL query contains commit and rollback query statements, the Data Integration Service must recreate the SQL procedure after each commit or rollback. To optimize performance, do not use transaction statements in an SQL transformation query.

Enable early selection or push-into filter optimization methods with the SQL transformation.

To increase performance, the consider enabling the early selection or push-into optimization method with the SQL transformation.

Early Selection Optimization with the SQL Transformation

The Data Integration Service can perform early selection optimization with an SQL transformation if the filter condition references only pass-through ports and the SQL transformation does not have side effects.

The SQL transformation has side effects under the following circumstances:

- The SQL query updates a database. The SQL query contains a statement such as as CREATE, DROP, INSERT, UPDATE, GRANT, or REVOKE.
- The transformation returns NULL rows for SELECT statements that return no results. The rows might contain pass-through port values, SQL error information, or the NUMRowsAffected field.

Enabling Early Selection Optimization with the SQL Transformation

Enable early selection optimization in the SQL transformation if the SQL transformation has no side effects.

1. Enable the **Return Database Output Only** option in the SQL transformation **Advanced Properties**.
2. Clear **Has Side Effects** in the transformation **Advanced Properties**.
3. If the transformation has a **NumAffectedRows** port, remove the port.

Push-Into Optimization with the SQL Transformation

With push-into optimization, the Data Integration Service pushes the filter logic from a Filter transformation in the mapping to the query in the SQL transformation.

Use the following rules and guidelines when you enable push-into optimization with the SQL transformation:

- The transformation SQL query must only contain SELECT statements.
- The transformation SQL query must be a valid subquery.
- The filter condition cannot refer to the SQL Error or NumRowsAffected fields.
- The names of the output ports must match the names of the columns in the SQL SELECT statement. When you reference an output port in a filter condition, the Data Integration Service pushes the corresponding port name to the SQL query. You can add aliases to the SQL if the columns in the query do not match the output port names. For example, `SELECT mycolname1 AS portname1, mycolname2 AS portname2`.
- The transformation cannot have side effects.

Push-Into Optimization with the SQL Transformation Example

An SQL transformation retrieves orders by customer ID. A Filter transformation that appears after the SQL transformation returns only the rows where the order amount is greater than 1000.

The Data Integration Service pushes the following filter into a SELECT statement in the SQL transformation:

```
orderAmount > 1000
```

Each statement in the SQL query becomes a separate subquery of the SELECT statement that contains the filter.

The following query statement shows the original query statement as a subquery in the SELECT statement:

```
SELECT <customerID>, <orderAmount>, … FROM (original query statements) ALIAS WHERE
<orderAmount> > 1000
```

If the SQL query has multiple statements, each statement is included in a separate subquery. The subquery has the same syntax, including the WHERE clause.

The ports *customerID* and *orderAmount*, are the names of the output ports in the SQL transformation. The subquery does not include pass-through ports, the SQL error, or the SQL statistics ports. If you push multiple filters into the SQL transformation, the WHERE clause contains all the filters.

Enabling Push-Into Optimization with the SQL Transformation

Enable push-into optimization by configuring properties on the SQL transformation **Advanced Properties** tab.

1. Clear **Has Side Effects**.
2. Enable **Return Database Output Only**.
3. Set **Max Out Row Count** to zero.
4. Enable push-into optimization.

Transformation Cache

When you run a mapping that uses an Aggregator, Joiner, Lookup, Rank, or Sorter transformation, the Data Integration Service creates caches in memory to process the transformation. If the Data Integration Service requires more space, it stores overflow values in cache files on disk.

Consider the following solution for transformation cache bottlenecks:

Configure the transformations to allocate enough space to store the cache in memory.

To improve the processing time for the Aggregator, Joiner, Lookup, Rank, or Sorter transformation, configure the transformations to allocate enough space to store the cache in memory. When you configure the amount of cache memory to be equal to or greater than what is required to cache the data and index, you increase performance by reducing the system I/O overhead. When the Data Integration Service writes cache files to disk, the processing time increases due to system I/O overhead.

By default, the Data Integration Service automatically configures the cache memory requirements at run time. After you run a mapping in auto cache mode, you can tune the cache sizes for the transformations. You analyze the transformation statistics in the mapping log to determine the cache sizes required to process the transformations in memory. When you configure the cache size to use the value specified in the mapping log, you can ensure that no allocated memory is wasted. However, the optimal cache size varies based on the size of the source data. Review the mapping logs after subsequent mapping runs to monitor changes to the cache size. If you configure a specific cache size for a reusable transformation, verify that the cache size is optimal for each use of the transformation in a mapping.

Transformation Error Elimination

In large numbers, transformation errors decrease the performance of the Data Integration Service. With each transformation error, the Data Integration Service pauses to determine the cause of the error and to remove the row causing the error from the data flow. The Data Integration Service typically writes the row into the mapping log file of the Data Integration Service logs.

Consider the following solutions for transformation error bottlenecks:

Check the mapping log file to see where the transformation errors occur and evaluate those transformation constraints.

Transformation errors occur when the Data Integration Service encounters conversion errors, conflicting mapping logic, and any condition set up as an error, such as null input. Check the mapping log file to see where the transformation errors occur. If the errors center around particular transformations, evaluate those transformation constraints.

Configure a lower tracing level.

If you need to run a mapping that generates a large number of transformation errors, it is possible to improve performance by setting a lower tracing level. However, this is not a recommended long-term solution to transformation errors.

Transformation Side Effects

A transformation has side effects if it returns rows and modifies an object, or if it interacts with other objects or functions. The transformation might modify a database, add to a total, raise an exception, write an email, or call other functions with side effects.

The Data Integration Service identifies which transformations have side effects before it optimizes a mapping. The Data Integration Service assumes that a transformation has side effects when it cannot determine if the transformation has side effects.

Transformations with side effects limit when the Data Integration Service can optimize the mapping. Early selection, branch pruning, global predicate optimization, and push-into optimization alter mapping results if the Data Integration Service applies them to a transformation that has side effects. With early selection and push-into optimization, filter logic is moved from a Filter transformation as close to the source as possible. If the filter occurs before the side effect function, the mapping results change.

For example, a transformation receives a customer ID and returns rows containing order information. The transformation also writes the orders to a file. If the Data Integration Service applies a filter optimization before it writes orders to the file, the file receives less rows than when the filter occurs later in the mapping. The transformation side effect is the function of writing the order records to a file.

The following transformations have side effects:

- SQL transformation, Web Service Consumer transformation, and Java transformation unless the side effects property is disabled.
- Transformations that call an ABORT() or ERROR() function, send email, or call a stored procedure.
- Transformations that write to files or databases.
- Transformations that maintain a count through a variable port. For example, COUNT=COUNT+1.

The SQL transformation, Web Service Consumer transformation, and Java transformation have side effects by default. If you configure the transformation to process rows without side effects, you can disable the **Has Side Effects** property in **Advanced Properties**. If the transformation does not have side effects, you can enable optimization by configuring additional properties in these transformations.

Web Service Consumer Transformation Optimization

The Web Service Consumer transformation can decrease performance when a mapping calls the web service multiple times.

Consider the following solution for Web Service Consumer transformation bottlenecks:

Configure the Web Service Consumer transformation to use cookie authentication.

The remote web service server tracks the web service consumer users based on the cookies. You can increase performance when a mapping calls a web service multiple times.

When you project the cookie port to a web service request message, the web service provider returns a cookie value in the response message. You can pass the cookie value to another transformation downstream in the mapping or you can save the cookie value in a file. When you save the cookie value in a file, you can configure the cookie as input to the Web Service Consumer transformation. You can project the cookie output port to any of the Web Service Consumer transformation output groups.

Enable early selection or push-into filter optimization methods with the Web Service Consumer transformation.

To increase performance, the Data Integration Service can apply the early selection or push-into optimization method with the Web Service Consumer transformation. To apply early selection optimization, the web service cannot have side effects and cannot treat faults as errors. To apply push-into optimization the web service cannot have side effects, cannot treat faults as errors, and the filter condition must reference pass-through ports.

The web service has a side effect if it performs other functions besides returning a response to the Web Service Consumer transformation. The web service has side effects if it modifies a database, writes to a file, writes emails, updates a count, or calls other web services with side effects.

Configure the Web Service Consumer transformation to send multiple requests in parallel.

To increase mapping performance, configure the Web Service Consumer transformation to send multiple requests in parallel. When you enable the Web Service Consumer transformation to create multiple concurrent connections to the web service, you can set the memory consumption limit and the number of concurrent connection limits.

Early Selection Optimization with the Web Service Consumer Transformation

When the Data Integration Service applies the early selection optimization method to the Web Service Consumer transformation, it moves filter conditions before the Web Service Consumer transformation in the mapping closer to the source.

Enabling Early Selection Optimization with the Web Service Consumer Transformation

Enable early selection optimization for the Web Service Consumer transformation if the transformation does not have side effects and it does not treat faults as errors.

1. Open the Web Service Consumer transformation **Advanced Properties** view.
2. Clear **Treat Fault as Error**.
3. Clear **Has Side Effects**.

Push-Into Optimization with the Web Service Consumer Transformation

You can configure push-into optimization with the Web Service Consumer transformation when the transformation is in a virtual table in an SQL data service.

The mapping calls the web service to retrieve a set of data or a subset of the data based on the statements in the end-user SQL query. The end-user SQL query contains an optional filter condition.

With push-into optimization, the Web Service Consumer transformation receives the filter value in a filter port. The filter port is an unconnected input port that you identify as a filter port when you configure push-into optimization. The filter port has a default value that ensures that the web service returns all rows if the end-user query contains no filter. The filter port is not a pass-through port.

Note: The filter field must be part of the root group in the web service request.

When you configure a filter port, you identify an output port in the Web Service Consumer transformation that receives the column data from the web service response. For example, if the filter port is an input port named EmployeeID, the output port from the response might be a port named EmployeeNum. The Developer tool needs to associate the input filter port and an output port in order to push the filter logic from the virtual table read to the web service consumer request. The input ports for a web service request are usually different than the output ports from the web service response.

The filter field cannot be a pass-through port. When you configure a filter port, the default value of the port changes to the value of the filter condition, so the pass-though output port value changes. A filter based on the output pass-through port returns unexpected results.

You can push multiple filter expressions to the Web Service Consumer transformation. Each filter condition must be the following format:

```
<Field> = <Constant>
```

The filter conditions must be joined by AND. You cannot join the conditions with an OR.

Push-Into Optimization with Web Service Consumer Transformation Example

An SQL data service returns orders for all customers or it returns orders for a specific customer based on the SQL query it receives from the user.

The data service contains a logical data object with the following components:

Customer table

An Oracle database table that contains customer information.

Web Service Consumer transformation

A transformation that calls a web service to retrieve the latest orders for customers. The Web Service Consumer transformation has input ports for customerID and orderNum. The transformation has pass-through ports that contain customer data that it receives from the Customer table. The orderNum port is the filter port and is not connected. orderNum has the default value "*". When the web service receives this value in the web service request, it returns all orders.

Orders virtual table

A virtual table that receives the customer and order data from the web service. The end-user queries this table. Orders contains a customer column, orderID column, and customer and order data.

The end-user passes the following SQL query to the SQL data service:

```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

The Data Integration Service splits the query to optimize the mapping. The Data Integration Service uses early selection optimization and moves the filter logic, `customer = 23`, to the Customer table read. The Data Integration Service uses push-into optimization and pushes the filter logic, `orderID = 56`, into the Web Service Consumer transformation filter port. The Web Service Consumer transformation retrieves ordersID 56 for customer 23.

Enabling Push-Into Optimization with the Web Service Consumer Transformation

Enable push-into optimization for the Web Service Consumer transformation if the transformation does not have side effects and it does not treat faults as errors.

1. Open the Web Service Consumer transformation **Advanced Properties** view.
2. Clear **Treat Fault as Error**.
3. Clear **Has Side Effects**.
4. Click the **Open** button in the **Push-Into Optimization** property.
5. Choose the filter port name in the Optimized Input dialog box.
   You can choose multiple filter ports.
6. Click the **Output** column.
7. For each filter port, choose the output port that contains the filtered column in the web service response.
8. Enter a default value for each filter port.

   Note: You cannot configure a default value for a Web Service Consumer port unless it is a filter port.

CHAPTER 5

Mapping Optimization

This chapter includes the following topics:

Mapping Optimization Overview

Optimize mappings to enable the Data Integration Service to transform and move data efficiently. Mapping-level optimization might take time to implement, but it can significantly boost mapping performance.

The optimization tasks apply to regular mappings, logical data object read and write mappings, virtual table mappings, and operation mappings. Focus on mapping-level optimization after you optimize the targets and sources.

To optimize a mapping, you can perform the following tasks:

- Configure the mapping with the least number of transformations and expressions to do the most amount of work possible.
- Delete unnecessary links between transformations to minimize the amount of data moved.
- Choose an optimizer level that determines which optimization methods the Data Integration Service can apply to the mapping. When the Data Integration Service optimizes a mapping, it attempts to reduce the amount of data to process. For example, the Data Integration Service can use early selection optimization to move a filter closer to the source. It can use the cost-based optimization method to change the join processing order.
- Choose a pushdown type to enable the Data Integration Service to determine whether it can pushdown partial or full transformation logic to the source database.
- Configure data object caching to enable the Data Integration Service cache logical data objects and access pre-built logical data objects when it runs a mapping. By default, the Data Integration Service extracts source data and builds required data objects when it runs a mapping. Mapping performance increases when the Data Integration Service can access pre-built data objects.

- Indicate if the SQL transformation, Web Service Consumer transformation, and the Java transformation do not have side effects when you configure these transformations. Some transformations have side effects that restrict optimization. For example, a transformation can have a side effect if the transformation writes to a file or database, adds to a count, raises an exception, or writes an email. In most cases, the Data Integration Service identifies which transformations have side effects that restrict optimization.

Related Topics:

- "Data Object Caching" on page 61

Optimization Methods

The Data Integration Service applies optimization methods to reduce the number of rows to process in the mapping. You can configure the optimizer level for the mapping to limit which optimization methods the Data Integration Service applies.

The Data Integration Service can apply the following optimization methods:

- Pushdown optimization
- Early projection optimization
- Early selection optimization
- Branch pruning optimization
- Push-into optimization
- Predicate optimization
- Global predicate optimization
- Cost-based optimization
- Dataship-join optimization
- Semi-join optimization

The Data Integration Service can apply multiple optimization methods to a mapping at the same time. For example, the Data Integration Service applies the early projection optimization, predicate optimization, global predicate optimization, branch pruning optimization, and early selection optimization or push-into optimization methods when you select the normal optimizer level.

Optimizer Levels

The Data Integration Service optimizes mappings based on the optimizer level that you configure. Configure the optimizer level when you want the mapping to use an optimizer level other than the normal. By default, each mapping uses the normal optimizer level.

You can choose one of the following optimizer levels:

None

The Data Integration Service does not apply any optimization.

Minimal

The Data Integration Service applies the early projection optimization method.

Normal

The Data Integration Service applies the early projection, early selection, branch pruning, push-into, global predicate optimization, and predicate optimization methods. Normal is the default optimization level.

Full

The Data Integration Service applies the cost-based, early projection, early selection, branch pruning, predicate, push-into, semi-join, and dataship-join optimization methods.

The Data Integration Service applies the normal optimizer level when you run a mapping from the **Run** menu or mapping editor in the Developer tool. When you run the mapping from the **Run** menu, the Data Integration Service applies the optimizer level in the mapping configuration. When you run the mapping from the command line, the Data Integration Service applies the optimization level from the mapping deployment properties in the application.

Note: The Data Integration Service does not apply the pushdown optimization method with an optimizer level. You can configure pushdown optimization for a mapping in the mapping run-time properties.

Filter Optimizations

Filter optimization increases performance by reducing the number of rows that pass through the mapping. The Data Integration Service can apply the early selection optimization or push-into optimization.

When the Data Integration Service applies a filter optimization method, it moves a filter as close to the source as possible in a mapping. If the Data Integration Service cannot move a filter before a transformation in a mapping, it might be able to push the filter logic into a transformation.

Early Projection Optimization Method

When the Data Integration Service applies the early projection optimization method, it identifies unused ports and removes the links between those ports.

The early projection optimization method improves performance by reducing the amount of data that the Data Integration Service moves across transformations. When the Data Integration Service processes a mapping, it moves the data from all connected ports in a mapping from one transformation to another. In large, complex mappings, or in mappings that use nested mapplets, some ports might not supply data to the target. The Data Integration Service identifies the ports that do not supply data to the target. After the Data Integration Service identifies unused ports, it removes the links between all unused ports from the mapping.

The Data Integration Service does not remove all links. For example, it does not remove the following links:

- Links connected to a transformation that has side effects.
- Links connected to transformations that call an ABORT() or ERROR() function, send email, or call a stored procedure.

If the Data Integration Service determines that all ports in a transformation are unused, it removes all transformation links except the link to the port with the least data. The Data Integration Service does not remove the unused transformation from the mapping.

The Developer tool enables this optimization method by default.

Predicate Optimization Method

When the Data Integration Service applies the predicate optimization method, it examines the predicate expressions that a mapping generates. It determines whether it can simpliify or rewrite the expressions to increase mapping performance.

When the Data Integration Service runs a mapping, it generates queries against the mapping sources and performs operations on the query results based on the mapping logic and the transformations within the mapping. The queries and operations often include predicate expressions. Predicate expressions represent the conditions that the data must satisfy. The filter and join conditions in Filter and Joiner transformations are examples of predicate expressions.

With the predicate optimization method, the Data Integration Service also attempts to apply predicate expressions as early as possible in the mapping to improve mapping performance.

The Data Integration Service infers relationships from by 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 Data Integration Service might be able to add "B>5" to the join condition.

The Data Integration Service applies the predicate optimization method with the early selection optimization method when it can apply both methods to a mapping. For example, when the Data Integration Service creates new filter conditions through the predicate optimization method, it also attempts to move them upstream in the mapping through the early selection method. Applying both optimization methods improves mapping performance when compared to applying either method alone.

The Data Integration Service applies the predicate optimization method if the application increases performance. The Data Integration Service does not apply this method if the application changes the mapping results or it decreases the mapping performance. The Data Integration Service applies this optimization method by default.

Predicate Optimization Rules and Guidelines

When the Data Integration Service rewrites a predicate expression, it applies mathematical logic to the expression to optimize it.

The Data Integration Service might perform any or all of the following actions:

- Identifies equivalent variables across predicate expressions in the mapping and generates simplified expressions based on the equivalencies.
- Identifies redundant predicates across predicate expressions in the mapping and removes them.
- Extracts subexpressions from disjunctive clauses and generates multiple, simplified expressions based on the subexpressions.
- Normalizes a predicate expression.
- Applies predicate expressions as early as possible in the mapping.

The Data Integration Service might not apply predicate optimization to a mapping when the mapping contains transformations with a datatype mismatch between connected ports.

The Data Integration Service might not apply predicate optimization to a transformation when any of the following conditions are true:

- The transformation contains explicit default values for connected ports.
- The transformation has side effects.
- The transformation does not allow predicates to be moved. For example, a transformation that has side effects might have this restriction.

The Developer tool enables the predicate optimization method by default.

Cost-Based Optimization Method

With cost-based optimization, the Data Integration Service evaluates a mapping, generates semantically equivalent mappings, and runs the mapping with the best possible performance. Cost-based optimization reduces run time for mappings that perform adjacent inner-join, and full-outer join operations.

Semantically equivalent mappings are mappings that perform identical functions and produce the same results. To generate semantically equivalent mappings, the Data Integration Service divides the original mapping into fragments. The Data Integration Service then determines which mapping fragments it can optimize.

During optimization, the Data Integration Service might add, remove, or reorder transformations within a fragment. The Data Integration Service verifies that the optimized fragments produce the same results as the original fragments and forms alternate mappings that use the optimized fragments.

The Data Integration Service can also apply a sorted merge join if it determines that the sorted merge join performance is better than the nested loop join performance. A sorted merge join uses sort order to arrange two data sets before performing the join. A nested loop join uses nested loops to join two data sets. The Data Integration Service might use the sorting information in the sources or create a Sorter transformation if the cost of sorting the data is less expensive then processing the nested loop join.

The Data Integration Service generates all or almost all of the mappings that are semantically equivalent to the original mapping. It uses profiling statistics or database statistics to compute the cost for the original mapping and each alternate mapping. Then, it identifies the mapping that runs most quickly. The Data Integration Service performs a validation check on the best alternate mapping to ensure that it is valid and produces the same results as the original mapping.

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

The Developer tool does not enable this method by default.

Dataship-Join Optimization Method

The dataship-join optimization method attempts to locate smaller data sets next to larger data sets to reduce join processing time. The Data Integration Service attempts to apply the dataship-join optimization method when there is a significant size difference between two tables.

For example, the Data Integration Service 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 Data Integration Service creates a temporary staging table in the database that contains the larger detail table. Then, the Data Integration Service 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 Data Integration Service performs the join operation, the Joiner transformation logic is processed in the database.

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

The Developer tool does not enable this method by default.

Dataship-Join Requirements for Increased Performance

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 the Data Integration Service finds it is faster to read all the data from the master and detail source without applying the dataship-join optimization method.

Dataship-Join Optimization Rules and Guidelines

The Data Integration Service can apply dataship-join optimization to a Joiner transformation if the transformation meets the following requirements:

- The join type must be normal, master outer, or detail outer.
- The detail pipeline must originate from a relational source.
- If the mapping uses target-based commits, the Joiner transformation scope must be All Input.
- The master and detail pipelines cannot share any transformation.
- The mapping cannot contain a branch between the detail source and the Joiner transformation.
- The Data Integration Service fails to apply the dataship-join optimization method if the database which contains the detail side of the join is an IBM DB2 database that does not support Unicode encoding.

Semi-Join Optimization Method

The semi-join optimization method attempts to reduce the amount of data extracted from the source by modifying join operations in the mapping.

The Data Integration Service 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 Data Integration Service attempts to decrease 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 Data Integration Service no longer reads unnecessary rows from the larger group source. The Data Integration Service 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 Data Integration Service performs analyses to determine whether semi-join optimization is possible and likely to be worthwhile. If the analyses determine that this method is likely to improve performance, the Data Integration Service applies it to the mapping. The Data Integration Service then reanalyzes the mapping to determine whether there are additional opportunities for semi-join optimization. It performs additional optimizations if appropriate.

The Developer tool does not enable this method by default.

Semi-Join Optimization Requirements for Increased Performance

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

- The Joiner transformation master source must have significantly fewer rows than the detail source.
- The detail source must be large enough to justify the optimization. When the Data Integration Service applies semi-join optimization, the method adds some overhead time to mapping processing. If the detail

source is small, the time required to apply the semi-join method might exceed the time required to process all rows in the detail source.

- The Data Integration Service must be able to obtain source row count statistics for a Joiner transformation in order to accurately compare the time requirements of the regular join operation against the semi-join operation.

Semi-Join Optimization Rules and Guidelines

The Data Integration Service can apply semi-join optimization to a Joiner transformation if the transformation meets the following requirements:

- The join type must be normal, master outer, or detail outer. The joiner transformation cannot perform a full outer join.
- The detail pipeline must originate from a relational source.
- The join condition must be a valid sort-merge-join condition. That is, each clause must be an equality of one master port and one detail port. If there are multiple clauses, they must be joined by AND.
- If the mapping does not use target-based commits, the Joiner transformation scope must be All Input.
- The master and detail pipelines cannot share any transformation.
- The mapping cannot contain a branch between the detail source and the Joiner transformation.

Early Selection Optimization Method

When the Data Integration Service applies the early selection optimization method, it splits, moves, or removes the Filter transformations in a mapping. It moves filters up the mapping closer to the source.

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

The early selection optimization method is enabled by default when you choose the normal or full optimizer level in the Developer tool. The Data Integration Service ignores early selection optimization if a transformation that appears before the Filter transformation has side effects. The Data Integration Service cannot determine if the SQL transformation, Web Service Consumer transformation, and Java transformation have side effects. You can configure early selection optimization for these transformations if they do not have side effects.

You can disable early selection if the optimization does not increase performance. The Data Integration Service enables this optimization method by default.

Global Predicate Optimization Method

When the Data Integration Service uses the global predicate optimization method, it removes those rows that can be filtered out as early as possible in the mapping. This reduces the number of rows that need to be processed by the mapping. The global predicate optimization method includes both 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 Data Integration Service might be able to add "B>5" to the join condition and move the Filter transformation closer to the source.

The global predicate optimization method applies predicate expressions more effectively than the predicate optimization method. The global predicate optimization method determines whether it can simplify or rewrite the expressions to increase mapping performance. It also attempts to apply 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 Data Integration Service uses the global predicate optimization method, it splits the filters, moves the filters closer to the source, or removes the filters in a mapping.

Branch Pruning Optimization Method

The Data Integration Service can apply the branch pruning optimization method to transformations that do not contribute any rows to the target in a mapping.

The Data Integration Service 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.

The Developer tool enables the branch pruning optimization method by default when you choose the normal or full optimizer level. You can disable branch pruning if the optimization does not increase performance by setting the optimizer level to minimal or none.

Push-Into Optimization Method

With push-into optimization, the Data Integration Service moves the Filter transformation logic into the transformation immediately upstream of the Filter transformation in the mapping. Push-into optimization increases performance by reducing the number of rows that pass through the mapping.

The Data Integration Service does not move filter logic into another transformation if the transformation has side effects. The Data Integration Service cannot determine if the SQL transformation, Web Service Consumer transformation, and Java transformation have side effects. However, you can configure the SQL transformation, Web Service Consumer transformation, and Java transformation for push-into optimization.

Pushdown Optimization

When the Data Integration Service applies pushdown optimization, it pushes transformation logic to the source database. The Data Integration Service translates the transformation logic into SQL queries and sends the SQL queries to the database. The source database runs the SQL queries to process the transformations.

Pushdown optimization increases mapping performance when the source database can process transformation logic faster than the Data Integration Service. The Data Integration Service also reads less data from the source.

The amount of transformation logic that the Data Integration Service pushes to the source database depends on the database, the transformation logic, and the mapping configuration. The Data Integration Service processes all transformation logic that it cannot push to a database.

When you apply pushdown optimization, the Data Integration Service analyzes the optimized mapping from the source to the target or until it reaches a downstream transformation that it cannot push to the source database. The Data Integration Service generates and executes a SELECT query for each source that has transformation logic pushed down. The Data Integration Service can also generate an INSERT query if the target was pushed to the database. The Data Integration Service reads the results of the SQL queries and processes the remaining transformations in the mapping.

The Data Integration Service applies pushdown optimization to a mapping when you select the pushdown type in the mapping run-time properties.

You can select the following pushdown types:

- None. Select no pushdown type for the mapping.
- Source. The Data Integration Service tries to push down as much transformation logic as it can to the source database.
- Full. The Data Integration Service pushes the full transformation logic to the source database.

You can also create a string parameter for the pushdown type and use the following parameter values:

- None
- Source
- Full

Full Pushdown Optimization

When the Data Integration Service applies full pushdown optimization, it pushes all the transformation logic in the mapping to the source database. You can configure full pushdown in the mapping run-time properties.

Full pushdown optimization is ideal when the source and target are in the same database or when transformations such as Aggregator and Filter transformations are processed in the source database and reduce the amount of data moved. For example, if a mapping contains a Teradata source and Teradata target, configure full pushdown optimization to push all the transformation logic for processing from a Teradata source database to a Teradata target database.

When you configure a mapping with an Update Strategy transformation for full pushdown, you must determine pushdown compatibility for the mapping.

The Data Integration Service can pushdown a mapping with an Update Strategy transformation in the following scenarios:

- If the target transformation connected to the Update Strategy transformation receives multiple rows that do not have the same key.
- If the target transformation connected to the Update Strategy transformation receives multiple rows with the same key that can be reordered.

The Data Integration Service cannot pushdown a mapping with an Update Strategy transformation in the following scenario:

- If the target transformation connected to the Update Strategy transformation receives multiple rows with the same key that cannot be reordered.

You can also use a pushdown compatibility parameter in the mapping. You can use the following parameter values:

- noMultipleRowsWithSameKeyOnTarget
- reorderAllowedForMultipleRowsWithSameKey
- reorderNotAllowedForRowsWithSameKey

The Data Integration Service can use full pushdown optimization for the following sources:

- Oracle
- IBM DB2
- Microsoft SQL Server
- Teradata
- Netezza
- Greenplum
- SAP HANA

Source Pushdown

When the Data Integration Service applies source pushdown, it analyzes the mapping from source to target or until it reaches a downstream transformation it cannot push to the source database.

The Data Integration Service generates and executes a SELECT statement based on the transformation logic for each transformation it can push to the database. Then, it reads the results of this SQL query and processes the remaining transformations.

You can configure a mapping to use source pushdown if the source and target reside in different databases. For example, if a mapping contains a Teradata source and an Oracle target, you can configure source pushdown to push some transformation logic for processing to the Teradata source.

Pushdown Optimization Rules and Guidelines

The Data Integration Service can push transformation logic to the source database.

The following rules and guidelines apply to pushdown optimization:

- The Data Integration Service can push Lookup and Joiner transformation logic to the source database if the sources are in the same database management system and they use identical connections.
- The Data integration Service cannot push transformation logic to a source that has a binary data type.
- The Data Integration Service disables pushdown optimization when you have an IBM DB2 data source and the column precision is between 28 to 31 digits for the Decimal data type.
- The Data Integration Service enables pushdown optimization for an SQL Data Service or a Web Service by default. You cannot disable pushdown optimization for an SQL Data Service or Web Service.
- The Data Integration Service cannot push an Aggregator transformation that contains an expression with aggregate and non-aggregate functions in a port that is not group by.

Single-Pass Reading

Single-pass reading allows you to populate multiple targets with one customized data object. Consider using single-pass reading if you have multiple mappings that use the same sources.

Consider the following solutions for single-pass reading bottlenecks:
Combine the transformation logic for each mapping in one mapping and use one customized data object for each source.

The Data Integration Service reads each source once and then sends the data into separate pipelines. A particular row can be used by all the pipelines, by any combination of pipelines, or by no pipelines.

For example, you have the Purchasing source table, and you use that source daily to perform an aggregation and a ranking. If you place the Aggregator and Rank transformations in separate mappings, you force the Data Integration Service to read the same source table twice. However, if you include the aggregation and ranking logic in one mapping with one source qualifier, the Data Integration Service reads the Purchasing source table once, and then sends the appropriate data to the separate pipelines.

Factor out common functions from mappings.

When changing mappings to take advantage of single-pass reading, you can optimize this feature by factoring out common functions from mappings. For example, if you need to subtract a percentage from the Price ports for both the Aggregator and Rank transformations, you can minimize work by subtracting the percentage before splitting the pipeline. You can use an Expression transformation to subtract the percentage, and then split the mapping after the transformation.

Filter Optimization

You can optimize mappings by filtering within a customized data object and by placing filters early in the mapping.

Consider the following solutions for filter bottlenecks:

Use a filter in a customized data object to remove the rows at the source.

If you filter rows from the mapping, you can improve efficiency by filtering early in the data flow. Use a filter in a customized data object to remove the rows at the source. The customized data object limits the row set extracted from a relational source.

If you cannot use a filter in the customized data object, use a Filter transformation and move it as close to the customized data object as possible to remove unnecessary data early in the data flow. The Filter transformation limits the row set sent to a target.

Use a filter in an Update Strategy transformation if you do not need to keep rejected rows.

To improve mapping performance, you can also use a Filter transformation to drop rejected rows from an Update Strategy transformation if you do not need to keep rejected rows.

Avoid complex expressions in filter conditions.

Avoid using complex expressions in filter conditions. To optimize Filter transformations, use simple integer or true/false expressions in the filter condition.

The Filter transformation filters data within a mapping. The Filter transformation filters rows from any type of source. The customized data object filters rows from relational sources. The Filter transformation filters rows from any type of source.

Datatype Conversion Optimization

You can increase performance by eliminating unnecessary datatype conversions. For example, if a mapping moves data from an Integer column to a Decimal column, then back to an Integer column, the unnecessary datatype conversion slows performance. Where possible, eliminate unnecessary datatype conversions from mappings.

Consider the following solutions for datatype conversion bottlenecks:

Use integer values in place of other datatypes when performing comparisons using Lookup and Filter transformations.

For example, many databases store U.S. ZIP code information as a Char or Varchar datatype. If you convert the zip code data to an Integer datatype, the lookup database stores the zip code 94303-1234 as 943031234. This helps increase the speed of the lookup comparisons based on zip code.

Convert the source dates to strings through port-to-port conversions to increase mapping performance.

You can either leave the ports in targets as strings or change the ports to Date/Time ports.

Error Tracing

To improve performance, reduce the number of log events generated by the Data Integration Service when it runs the mapping. Improve mapping performance by updating the mapping optimizer level through the mapping configuration or mapping deployment properties. Use the cost-based optimization method to optimize mappings.

Consider the following solutions for error tracing bottlenecks:

Set the tracing level in the mapping properties to Terse

If a mapping contains a large number of transformation errors, and you do not need to correct them, set the tracing level in the mapping properties to Terse. At this tracing level, the Data Integration Service does not write error messages or row-level information for reject data.

If you need to debug the mapping and you set the tracing level to Verbose, you may experience significant performance degradation when you run the mapping. Do not use Verbose tracing when you tune performance. The mapping tracing level overrides any transformation-specific tracing levels within the mapping. This is not recommended as a long-term response to high levels of transformation errors.

Change the optimizer level for the mapping.

If a mapping takes an excessive amount of time to run, you might want to change the optimizer level for the mapping. The optimizer level determines which optimization methods the Data Integration Service applies to the mapping at run-time.

You set the optimizer level for a mapping in the mapping configuration or mapping deployment properties. The Data Integration Service applies different optimizer levels to the mapping depending on how you run the mapping.

Use the cost-based optimization method.

The cost-based optimization method causes the Data Integration Service to evaluate a mapping, generate semantically equivalent mappings, and run the mapping with the best performance. This method is most effective for mappings that contain multiple Joiner transformations. It reduces run time for mappings that perform adjacent, unsorted, inner-join operations.

Semantically equivalent mappings are mappings that perform identical functions and produce the same results. To generate semantically equivalent mappings, the Data Integration Service divides the original mapping into fragments. The Data Integration Service then determines which mapping fragments it can optimize.

CHAPTER 6

Partitioned Mapping Optimization

This chapter includes the following topics:

Partitioned Mapping Optimization Overview

If you have the partitioning option, you can enable the Data Integration Service to maximize parallelism when it runs mappings. When you maximize parallelism, the Data Integration Service dynamically divides the underlying data into partitions and processes all of the partitions concurrently.

If mappings process large data sets or contain transformations that perform complicated calculations, the mappings can take a long time to process and can cause low data throughput. When you enable partitioning for these mappings, the Data Integration Service uses additional threads to process the mapping.

You can optimize the performance of partitioned mappings by performing the following tasks:

- Use multiple CPUs on the nodes that run mappings.
- Increase the maximum parallelism value for the Data Integration Service.
- Configure properties on flat file data objects.
- Configure relational databases to optimize partitioning.
- Configure properties on transformations.

Use Multiple CPUs

Increasing the number of processing threads increases the load on the nodes that run mappings. If the nodes contain ample CPU bandwidth, concurrently processing rows of data in a mapping can optimize mapping performance.

The Data Integration Service can use multiple CPUs to process a mapping that contains multiple partitions. The number of CPUs that the service uses depends on factors such as the number of partition points, the

number of threads created for each pipeline stage, and the amount of resources required to process the mapping. A simple mapping runs faster in two partitions, but typically requires twice the amount of CPU than when the mapping runs in a single partition.

Increase the Maximum Parallelism Value

Maximum parallelism determines the maximum number of parallel threads that can process a single pipeline stage. Configure the **Maximum Parallelism** property for the Data Integration Service based on the available hardware resources. When you increase the maximum parallelism value, you might decrease the amount of processing time.

Consider the following guidelines when you increase the maximum parallelism value:

Increase the value based on the number of available CPUs.

Increase the maximum parallelism value based on the number of CPUs available on the nodes where mappings run. When you increase the maximum parallelism value, the Data Integration Service uses more threads to run the mapping and leverages more CPUs. A simple mapping runs faster in two partitions, but typically requires twice the amount of CPU than when the mapping runs in a single partition.

Consider the total number of processing threads.

Consider the total number of processing threads when setting the maximum parallelism value. If a complex mapping results in multiple additional partition points, the Data Integration Service might use more processing threads than the CPU can handle.

The total number of processing threads is equal to the maximum parallelism value.

Consider the other jobs that the Data Integration Service must run.

If you configure maximum parallelism such that each mapping uses a large number of threads, fewer threads are available for the Data Integration Service to run additional jobs.

Optionally change the value for a mapping.

By default, the maximum parallelism for each mapping is set to Auto. Each mapping uses the maximum parallelism value defined for the Data Integration Service.

In the Developer tool, developers can change the maximum parallelism value in the mapping run-time properties to define a maximum value for a particular mapping. When maximum parallelism is set to different integer values for the Data Integration Service and the mapping, the Data Integration Service uses the minimum value of the two.

Optimize Flat Files for Partitioning

When a mapping that is enabled for partitioning reads from a flat file source or writes to a flat file target, the Data Integration Service can use multiple threads to read from or to write to the flat file.

Optimize Flat File Sources for Partitioning

To achieve optimal performance when using multiple threads to read from a flat file, configure the flat file data object to optimize throughput instead of preserving row order.

Consider the following solution to reduce bottlenecks for partitioned flat file sources:

Configure concurrent read partitioning for the flat file data object to optimize throughput.

In the flat file data object advanced properties, set the **Concurrent Read Partitioning** property to optimize throughput. When you optimize throughput, the Data Integration Service does not preserve row order because it does not read the rows in the file or file list sequentially.

Optimize Flat File Targets for Partitioning

To achieve optimal performance when using multiple threads to write to a flat file, configure partitions to write the target output to separate files and configure multiple target directories.

Consider the following solutions to reduce bottlenecks for partitioned flat file targets:

Configure partitions to write the target output to separate files.

In the flat file data object advanced properties, set the **Merge Type** property to **No merge**. The Data Integration Service concurrently writes the target output to a separate file for each partition. If you require merged target data, the concurrent merge type optimizes performance more than the sequential merge type.

Configure multiple target directories.

When multiple threads write to a single directory, the mapping might encounter a bottleneck due to input/output (I/O) contention. An I/O contention can occur when threads write data to the file system at the same time. When you configure multiple directories, the Data Integration Service determines the output directory for each thread in a round-robin fashion.

Configure the output file directories in the advanced properties for the flat file data object. Use the default TargetDir system parameter value if an administrator entered multiple directories separated by semicolons for the **Target Directory** property for the Data Integration Service in the Administrator tool. Or, you can enter a different value to configure multiple output file directories specific to the flat file data object.

Optimize Relational Databases for Partitioning

When a mapping that is enabled for partitioning reads from or writes to an IBM DB2 for LUW or an Oracle relational database, the Data Integration Service can use multiple threads to read the relational source or to write to the relational target.

To optimize performance when using multiple threads to read from or write to a DB2 for LUW or Oracle relational database, you can partition the source and target tables.

Note: If a mapping reads from or writes to a relational database other than DB2 for LUW or Oracle, the Data Integration Service uses one reader thread or one writer thread.

Optimize the Source Database for Partitioning

To achieve optimal performance when using multiple threads to read from a DB2 for LUW or an Oracle source database, verify that the source table is partitioned and is configured to accept parallel queries.

To optimize the source database for partitioning, perform the following tasks:

Add database partitions to the source.

Add database partitions to the relational source to increase the speed of the Data Integration Service query that reads the source. If the source does not have database partitions, the Data Integration Service uses one thread to read from the source.

Enable parallel queries.

Relational databases might have options that enable parallel queries to the database. Refer to the database documentation for these options. If these options are not enabled, the Data Integration Service runs multiple partition SELECT statements serially.

Separate data into different tablespaces.

Each database provides an option to separate the data into different tablespaces. Each tablespace can refer to a unique file system, which prevents any I/O contention across partitions.

Increase the maximum number of sessions allowed to the database.

The Data Integration Service creates a separate connection to the source database for each partition. Increase the maximum number of allowed sessions so that the database can handle a larger number of concurrent connections.

Optimize the Target Database for Partitioning

To achieve optimal performance when using multiple threads to write to a DB2 for LUW or an Oracle target database, verify that the target table is partitioned and is configured to insert rows in parallel.

To optimize the target database for partitioning, perform the following tasks:

Add database partitions to a DB2 for LUW target.

The Data Integration Service can use multiple threads to write to a DB2 for LUW target that does not have database partitions. However, you can optimize load performance when the target has database partitions. In this case, each writer thread connects to the DB2 for LUW node that contains the database partition. Because the writer threads connect to different DB2 for LUW nodes instead of all threads connecting to the single master node, performance increases.

Enable parallel inserts.

Relational databases might have options that enable parallel inserts to the database. Refer to the database documentation for these options. For example, set the db_writer_processes option in an Oracle database and the max_agents option in a DB2 for LUW database to enable parallel inserts.

Separate data into different tablespaces.

Each database provides an option to separate the data into different tablespaces. Each tablespace can refer to a unique file system, which prevents any I/O contention across partitions.

Increase the maximum number of sessions allowed to the database.

The Data Integration Service creates a separate connection to the target database for each partition. Increase the maximum number of allowed sessions so that the database can handle a larger number of concurrent connections.

Set options to enhance database scalability.

Relational databases might have options that enhance scalability. For example, disable archive logging and timed statistics in an Oracle database to enhance scalability.

Optimize Transformations for Partitioning

When the Data Integration Service uses multiple threads to run an Aggregator, Joiner, Rank, or Sorter transformation, the service uses cache partitioning to divide the cache size across the threads. To optimize performance for cache partitioning, configure multiple cache directories.

Note: A Lookup transformation can only use a single cache directory.

Consider the following solution to reduce bottlenecks for partitioned Aggregator, Joiner, Rank, and Sorter transformations:

Configure multiple cache directories.

Cache partitioning creates a separate cache for each partition that processes an Aggregator, Joiner, Rank, or Sorter transformation. During cache partitioning, each partition stores different data in a separate cache. Each cache contains the rows needed by that partition. Cache partitioning optimizes mapping performance because each thread queries a separate cache in parallel.

If the cache size is smaller than the amount of memory required to run the transformation, transformation threads write to the cache directory to store overflow values in cache files. When multiple threads write to a single directory, the mapping might encounter a bottleneck due to I/O contention. An I/O contention can occur when threads write data to the file system at the same time. When you configure multiple cache directories, the Data Integration Service determines the cache directory for each transformation thread in a round-robin fashion.

In an Aggregator, Joiner, or Rank transformation, configure the cache directories in the **Cache Directory** advanced property. Use the default CacheDir system parameter value if an administrator entered multiple directories separated by semicolons for the **Cache Directory** property for the Data Integration Service in the Administrator tool. Or, you can enter a different value to configure multiple cache directories specific to the transformation.

In a Sorter transformation, configure the cache directories in the **Work Directory** advanced property. Use the default TempDir system parameter value if an administrator entered multiple directories separated by semicolons for the **Temporary Directories** property for the Data Integration Service in the Administrator tool. Or, you can enter a different value to configure multiple cache directories specific to the transformation.

CHAPTER 7

Run-time Optimization

This chapter includes the following topics:

Run-time Optimization Overview

Enable performance features and tune Data Integration Service properties to optimize mapping performance.

Use the following optimization techniques in the Administrator tool to get the best performance results based on your requirements:

- Optimize application service processes.
- Configure monitoring statistics to monitor system bottlenecks.
- Allocate memory for optimal system performance.
- Configure data object caching.
- Optimize the system to avoid system delays and slow disk access.

Application Service Optimization

Optimize the application service process when performance is affected. You can optimize the Analyst Service, Data Integration Service, and the Model Repository Service.

Analyst Service Optimization

Tune the Analyst Service to optimize performance. You can configure the Analyst Service process property for memory, minimize network latency, and configure the Analyst tool flat file upload settings to improve service performance.

Consider the following solutions for Analyst Service bottlenecks:

Configure the Analyst tool to connect to a network path location to upload flat files greater than 10 MB.

The Analyst Service process performance can decrease when analysts upload flat files greater than 10 MB to the Informatica Installation directory on the machine on which the Analyst tool runs. This can affect both disk space and network performance.

Upload flat files smaller than 10 MB to the Informatica Installation directory from the Analyst tool.

The Analyst Service process performance can decrease if analysts upload flat files greater than 10 MB to the Informatica Installation directory from the Analyst tool. This can affect both disk space and network performance.

Increase the Maximum Heap Size property for the Analyst Service process.

The Analyst Service process can consume large amounts of memory while processing a large number of concurrently logged in users. This can cause a large number of network connections to be open between the Analyst Service and other services such as the Data Integration Service or the Model Repository Service.

Use the Administrator tool to configure the Maximum Heap Size property to a larger value in the Advanced Properties for the Analyst Service process.

Export large mapping specifications to a table or export to a flat file and truncate the file.

The Analyst Service process can have a performance impact when analysts export large mapping specifications as flat files from the Analyst tool.

Data Integration Service Optimization

Tune the Data Integration Service process to improve service performance. You can configure the Data Integration Service process properties for memory. You can configure each web service and SQL data service that runs on a Data Integration Service to handle concurrent requests.

Consider the following solutions for Data Integration Service bottlenecks:

Configure the Maximum Heap Size property for the Data Integration Service process.

The Data Integration Service can consume large amounts of memory while processing SQL data services and web services.

Use the Administrator tool to configure the Maximum Heap Size property to a larger value in the Advanced Properties for the Data Integration Service process.

Configure the web service DTM Keep Alive Time property for the Data Integration Service.

The Data Integration Service consumes system resources to spawn a DTM instance for each web service request. Configure the Data Integration Service to use one DTM instance to process more than one web service request.

Use the Administrator tool to configure the web service DTM Keep Alive Time property for the Data Integration Service.

Configure the execution options in the Data Integration process properties and the web service and SQL data service properties for concurrent requests.

The Data Integration Service, each SQL data service, and each web service that runs on the Data Integration Service consumes system and memory resources for each concurrent request.

To configure the number of concurrent requests that the Data Integration Service, each SQL data service, and each web service can accept, configure the Data Integration Service properties and the web service properties.

Use the Administrator tool to configure the following options and properties for the Data Integration Service, web service and SQL data service:

- Configure the execution options for the Data Integration Service.
- Configure the Maximum # of Concurrent Connections property for each SQL data service in the SQL properties for the Data Integration Service process.
- Configure the Maximum Backlog Request and the Maximum Concurrent Requests properties for each web service in the HTTP configuration properties for the Data Integration Service process.

Turn off the web service trace level.

The number of web service log files that the Data Integration Service writes and maintains can decrease performance.

Use the Administrator tool to configure the web service trace level to reduce the amount of web service run-time log files that the Data Integration Service stores on disk.

Model Repository Service Optimization

Tune the Model Repository Service to improve performance. You can configure the Model Repository Service process property for memory and minimize network latency.

Consider the following solutions for Model Repository Service bottlenecks:

Host the Model repository database on the same machine as the Model Repository Service.

The Model Repository service process performance can be affected if the Model repository database is hosted on a remote server. Model Repository Service operations that require communication between the Model repository and the Model Repository Service on a high-latency network could slow down the Model Repository Service performance.

Increase the Maximum Heap Size property for the Model Repository Service process.

The Model Repository Service process can consume large amounts of memory while processing a large number of concurrently logged in users. This can cause a large number of network connections to be open between the Model Repository Service and other services such as the Data Integration Service or the Analyst Service.

Use the Administrator tool to configure the Maximum Heap Size property to a larger value in the Advanced Properties for the Model Repository Service process.

Monitoring Statistics

Monitoring is a domain function that the Service Manager performs. The Service Manager stores the monitoring configuration in the Model repository. Use the Monitor tab in the Administrator tool to monitor system bottlenecks such as the total number of running, failed, canceled, and completed jobs that run on a selected service.

Consider the following solution for monitoring statistics bottlenecks:

Configure the domain to set up monitoring.

When you set up monitoring, the Data Integration Service stores persisted statistics and monitoring reports in the Model repository. Persisted statistics are historical information about integration objects that previously ran. The monitoring reports show key metrics about an integration object.

Configure monitoring settings for the domain to specify the Model repository that stores the run-time statistics about objects deployed to Data Integration Services. Monitoring settings apply to all Data Integration Services that in the domain and can affect service performance.

The following table describes monitoring settings that can affect services performance:

Option	Description
Preserve Summary Historical Data	Number of days that the Model repository saves averaged data. If purging is disabled, then the Model repository saves the data indefinitely. Default is 180. Minimum is 0. Maximum is 366.
Preserve Detailed Historical Data	Number of days that the Model repository saves per-minute data. If purging is disabled, then the Model repository saves the data indefinitely. Default is 14. Minimum is 1. Maximum is 14.
Purge Statistics Every	Interval, in days, at which the Model Repository Service purges data that is older than the values configured in the **Preserve Historical Data** option. Default is 1 day.
Days At	Time of day when the Model Repository Service purges statistics. Default is 1:00 a.m.
Maximum Number of Sortable Records	Maximum number of records that can be sorted in the **Monitor** tab. If the number of records on the **Monitor** tab is greater than this value, then you can only sort by **Start Time** and **End Time**. Default is 3,000.
Maximum Delay for Update Notifications	Maximum time, in seconds, that the Data Integration Service buffers statistics before it stores them in the Model repository and displays them in the **Monitor** tab. If the Data Integration Service shuts down unexpectedly before it stores the statistics in the Model repository, then the statistics are lost. Default is 10.
Show Milliseconds	Include milliseconds for date and time fields in the **Monitor** tab.

Memory Allocation

To optimize mapping performance, configure memory properties for the Data Integration Service in the Administrator tool.

The following table describes the maximum memory per request property for the Mapping Service Module:

Property	Description
Maximum Memory Per Request	The behavior of Maximum Memory Per Request depends on the following Data Integration Service configurations: - The service runs jobs in separate local or remote processes, or the service property Maximum Memory Size is 0 (default). In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to all transformations that use auto cache mode in a single request. The service allocates memory separately to transformations that have a specific cache size. The total memory used by the request can exceed the value of Maximum Memory Per Request. - The service runs jobs in the Data Integration Service process, and the service property Maximum Memory Size is greater than 0. In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to a single request. The total memory used by the request cannot exceed the value of Maximum Memory Per Request. Default is 536,870,912.

The following table describes the execution options for the Data Integration Service:

Property	Description
Maximum Memory Size	Maximum amount of memory, in bytes, that the Data Integration Service can allocate for running all requests concurrently when the service runs jobs in the Data Integration Service process. When the Data Integration Service runs jobs in separate local or remote processes, the service ignores this value. If you do not want to limit the amount of memory the Data Integration Service can allocate, set this property to 0. If the value is greater than 0, the Data Integration Service uses the property to calculate the maximum total memory allowed for running all requests concurrently. The Data Integration Service calculates the maximum total memory as follows: Maximum Memory Size + Maximum Heap Size + memory required for loading program components Default is 0. **Note:** If you run profiles or data quality mappings, set this property to 0.

The following table describes the maximum heap size property for the Data Integration Service process:

Property	Description
Maximum Heap Size	Amount of RAM allocated to the Java Virtual Machine (JVM) that runs the Data Integration Service. Use this property to increase the performance. Append one of the following letters to the value to specify the units: - b for bytes. - k for kilobytes. - m for megabytes. - g for gigabytes. Default is 640 megabytes. **Note:** Consider increasing the heap size when the Data Integration Service needs to process large amounts of data.

Data Object Caching

The Data Integration Service uses data object caching to access pre-built logical data objects and virtual tables. Enable data object caching to increase performance for mappings, SQL data service queries, and web service requests that include logical data objects and virtual tables.

By default, the Data Integration Service extracts source data and builds required data objects when it runs a mapping, SQL data service query, or a web service request. When you enable data object caching, the Data Integration Service can use cached logical data objects and virtual tables.

Perform the following steps to configure data object caching for logical data objects and virtual tables in an application:

1. Configure the data object cache database connection in the cache properties for the Data Integration Service.
2. Enable caching in the properties of logical data objects or virtual tables in an application.

By default, the Data Object Cache Manager component of the Data Integration Service manages the cache tables for logical data objects and virtual tables in the data object cache database. When the Data Object Cache Manager manages the cache, it inserts all data into the cache tables with each refresh. If you want to incrementally update the cache tables, you can choose to manage the cache tables yourself using a database client or other external tool. After enabling data object caching, you can configure a logical data object or virtual table to use a user-managed cache table.

To use the Timestamp with Time Zone data type and to enable data object caching for IBM DB2 or for Microsoft SQL Server, set the date time format of the deployed mapping to the "YYYY-MM-DD HH24:MI:SS" format. The Data Integration Service writes the data up to seconds.

Data Types for Cache Tables

The Data Integration Service uses data from cache tables when it processes mappings, SQL data service queries, and web service requests that contain cached objects. The cache table data types that the Data Integration Service expects can differ from the cached object data types.

The Data Object Cache Manager creates the cache tables with the data types that the Data Integration Service expects. If you use user-managed cache tables, verify that the cache tables use the data types that the Data Integration Service expects.

Virtual Table Cache Data Types

The following table lists the cache table data types for virtual tables:

Virtual Table Data Type	IBM DB2	Microsoft SQL Server	Oracle
Char	Vargraphic Dbclob, for precision greater than 32672	Nvarchar Ntext, for precision greater than 4000	Nvarchar2 Nclob, for precision greater than 2000
Bigint	Bigint	Bigint	Number
Boolean	Integer	Int	Number
Date	Time stamp	Datetime2	Time stamp
Double	Double	Float	Time stamp
Decimal	Decimal	Decimal	Number
Int	Integer	Int	Number
Time	Time stamp	Datetime2	Time stamp
Time stamp	Time stamp	Datetime2	Time stamp
Varbinary	Blob	Binary Image, for precision greater than 8000	Raw Blob, for precision greater than 2000
Varchar	Vargraphic Dbclob, for precision greater than 32672	Nvarchar Ntext, for precision greater than 4000	Nvarchar2 Nclob, for precision greater than 2000

Logical Data Object Cache Data Types

The following table lists the cache table data types for logical data objects:

Logical Data Object Data Type	DB2	Microsoft SQL Server	Oracle
Bigint	Bigint	Bigint	Number
Binary	Blob	Binary Image, for precision greater than 8000	Raw Blob, for precision greater than 2000
Date/time	Time stamp	Datetime2	Time stamp
Double	Double	Float	Number
Decimal	Decimal	Decimal	Number
Integer	Integer	Int	Number

Logical Data Object Data Type	DB2	Microsoft SQL Server	Oracle
String	Vargraphic Dbclob, for precision greater than 32672	Nvarchar Ntext, for precision greater than 4000	Nvarchar2 Nclob, for precision greater than 2000
Text	Vargraphic Dbclob, for precision greater than 32672	Nvarchar Ntext, for precision greater than 4000	Nvarchar2 Nclob, for precision greater than 2000

Data Object Cache Optimization

Cache performance depends on the performance of the cache database and the configuration of objects within mappings, SQL data services, and web services.

Consider the following solutions to increase cache performance:

Optimize the cache database.

Optimal performance for the cache depends on the speed and performance of the cache database and the cache size. Configure the cache size within the cache database.

Because the Data Object Cache Manager must maintain the old cache for a refresh operation, the cache must be large enough to store two sets of data. Use the following formula to estimate the required minimum cache size:

```
2 * average data object size * number of data objects
```

For example, you want to cache 20 logical data objects and 10 virtual tables. If your average object size is 15 MB, then the required cache size is 2 * 15 MB * (20 + 10) = 900 MB.

Cache tables are read-only. End users cannot update the cache tables with SQL commands.

Define primary keys and foreign keys for logical data objects.

When the Data Integration Service generates cache for logical data objects with keys, it creates indexes. The indexes can increase the performance of queries on the cache database.

Cache logical data objects that you join in a mapping.

When you join cached logical data objects, the Data Integration Service can push down the Joiner transformation logic to the cache database even when the source data originates from different databases.

Generate index cache based on columns in a logical data object or virtual table.

Configure the Data Integration Service to generate an index cache based on columns in logical data objects or virtual tables. The index can increase the performance of queries on the cache database.

System Optimization

Often performance slows because the mapping relies on inefficient connections or an overloaded Data Integration Service process system. System delays can also be caused by routers, switches, network protocols, and usage by many users.

Slow disk access on source and target databases, source and target file systems, and nodes in the domain can slow mapping performance. Have the system administrator evaluate the hard disks on the machines.

Consider the following solutions for system optimization bottlenecks:

Improve network speed.

Slow network connections can slow mapping performance. Have the system administrator determine if the network runs at an optimal speed. Decrease the number of network hops between the Data Integration Service process and databases.

Use multiple CPUs.

You can use multiple CPUs to run multiple mappings in parallel.

Reduce paging.

When an operating system runs out of physical memory, it starts paging to disk to free physical memory. Configure the physical memory for the Data Integration Service process machine to minimize paging to disk.

Use processor binding.

In a multi-processor UNIX environment, the Data Integration Service may use a large amount of system resources. Use processor binding to control processor usage by the Integration Service process. Also, if the source and target database are on the same machine, use processor binding to limit the resources used by the database.

CHAPTER 8

SQL Data Service Optimization

This chapter includes the following topics:

SQL Data Service Optimization Overview

You can optimize SQL data services to improve performance when end users run SQL queries against them using third-party client tools. If an SQL data service uses a virtual table mapping, you can optimize the sources, transformations, and the mapping.

Use the following optimization techniques to optimize an SQL data service:

- Optimize third-party client tools.
- Configure the SQL data service optimizer level.
- Configure SQL data service properties for concurrency and memory for a Data Integration process.
- Configure data object caching for the SQL data service.
- Configure result set caching for the SQL data service.
- Configure constraints for virtual tables in the SQL data service.

Related Topics:

- "Data Object Caching" on page 61

Third-party Client Tool Optimization

Third-party client tools can affect performance when processing and running SQL queries against an SQL data service. Optimize the third-party client tools that end users can use to run SQL queries against an SQL data service.

Consider the following solutions for third-party client tool bottlenecks:

Send large query results to a file on disk.

A third-party client tool can affect performance if it displays large query results on the console window.

Configure the third-party client tool to disable encryption.

A third-party client tool can affect performance if it encrypts data while fetching or displaying query results.

Configure the third-party client tool to previously fetch a set of rows.

A third-party client tool can affect performance if it fetches single rows at a time.

Configure the third-party client tool to disable the option to read contents from the table when it is first loaded.

A third-party client tool can affect performance if the datatype settings for the BLOB and CLOB datatypes are configured to read contents from the table when it is first loaded if the BLOB and CLOB datatypes are not used in the query.

Configure the third-party client tool to use the default format and conversion settings for Date, Time, and Timestamp.

A third-party client tool can affect performance if the Date, Time, and Timestamp format and conversion settings are set to a user-specified format instead of the default format.

Disable the debug option or set it to no debug.

A third-party client tool can affect performance if the debug option to run the query is set to trace. This can slow down performance as the third-party client tool writes more log messages to the debug file while processing the query.

SQL Data Service Optimizer Levels

The Data Integration Service optimizes SQL data services based on the optimizer level that you configure. Configure the optimizer level when you want the SQL data service to use an optimizer level other than normal. By default, each SQL data service uses the normal optimizer level.

To understand how the optimizer level creates an optimized query for an SQL data service, view the query plan for an SQL Data Service. When you view the query plan, the Developer tool displays a graphical representation of the optimized query based on the optimizer level and a graphical representation of the original query.

You can configure the following optimizer levels:

None

The Data Integration Service does not apply any optimization.

Minimal

The Data Integration Service applies the early projection optimization method.

Normal

The Data Integration Service applies the early projection, early selection, branch pruning, push-into, global predicate optimization, and predicate optimization methods. Normal is the default optimization level.

Full

The Data Integration Service applies the cost-based, early projection, early selection, branch pruning, predicate, push-into, semi-join, and dataship-join optimization methods.

You can use one or more of the following methods to configure the optimizer level for an SQL data service:

- Configure the optimizer level for data preview of SQL data services.
- Configure the optimization level for deployed SQL data services.
- Configure the optimizer level in the connection string of queries that you run against a deployed SQL data services.

Configuring the SQL Data Service Optimizer Level for Data Preview

Configure the optimizer level that the Data Integration Service uses to execute SQL queries when you preview the output of a SQL data service.

1. In the Developer tool, click **Run** > **Open Run Dialog**.

   The **Run** dialog box appears.
2. Click **Data Viewer Configuration**.
3. Click the **New** button.
4. Enter a name for the data viewer configuration.
5. Click the **Advanced** tab.
6. Select an optimizer level.
7. Click **Apply**.
8. Click **Close**

   The Developer tool creates the data viewer configuration.

Configuring the Optimizer Level for Deployed SQL Data Services

Configure the optimizer level that the Data Integration Services uses to execute SQL queries against a deployed SQL data service. You can choose to override the optimizer level for a single query by configuring the optimizer level in the SQL data service connection.

1. In the Administrator tool, select a Data Integration Service.
2. Click the **Applications** view.
3. Expand the application that contains the SQL data service for which you want to configure the optimizer level.

4. Select the SQL data service and edit the following property:

Property	Description
Optimization Level	The optimizer level that the Data Integration Service applies to the object. Enter the numeric value that is associated with the optimizer level that you want to configure. You can enter one of the following numeric values: - 0. The Data Integration Service does not apply optimization. - 1. The Data Integration Service applies the early projection optimization method. - 2. The Data Integration Service applies the early projection, early selection, push-into, and predicate optimization methods. - 3. The Data Integration Service applies the cost-based, early projection, early selection, push-into, predicate, and semi-join optimization methods.

5. To override optimizer level that the Data Integration Services uses to execute a query, append the following entry to the JDBC URL or ODBC connection string: `SQLDataServiceOptions.optimizeLevel=<numeric_optimizer_level>`.

SQL Data Service Query Plan

When you view the query plan for an SQL data service, you view the graphical representation of the original query and the graphical representation of the optimized query. The graphical representation describes how the Data Integration Service processes the query. It includes the transformations and the order which the Data Integration Services processes each transformation.

The Developer tool uses the optimizer level that you set in the Developer tool to generate the optimized query. The optimized query displays the query as the Data Integration Service runs it.

For example, you want to query the CUSTOMERS virtual table in an SQL data service. In the **Data Viewer** view, you choose the default data viewer configuration settings, which sets the optimizer level for the query to normal.

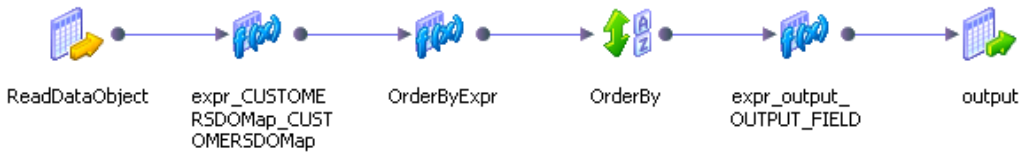
You enter the following query in the **Data Viewer** view:

```
select * from CUSTOMERS where CUSTOMER_ID > 150000 order by LAST_NAME
```

When you view the SQL query plan, the Developer tool displays the following graphical representation of the query:

The non-optimized view displays the query that you enter. The Developer tool displays the WHERE clause as a Filter transformation and the ORDER BY clause as a Sorter transformation. The Developer tool uses the pass-through Expression transformation to rename ports.

When you view the optimized query, the Developer tool displays the following graphical representation of the query:



The optimized view displays the query that the Data Integration Service runs. Because the optimizer level is normal, the Data Integration Service pushes the filter condition to the source data object. Pushing the filter condition increases query performance because it reduces the number of rows that the Data Integration Service reads from the source data object. Similar to the non-optimized query, the Developer tool displays the ORDER BY clause as a Sorter transformation. It uses pass-through Expression transformations to enforce the datatypes that you specify in the logical transformations.

Viewing an SQL Query Plan

Display the SQL query plan to view a mapping-like representation of the SQL query you enter when you preview virtual table data.

1. Open an SQL data service that contains at least one virtual table.
2. Click the **Data Viewer** view.
3. Enter an SQL query in the **Input** window.
4. Optionally, select a data viewer configuration that contains the optimizer level you want to apply to the query.
5. Click **Show Query Plan**.

   The Developer tool displays the SQL query plan for the query as you entered it on the **Non-Optimized** tab.
6. To view the optimized query, click the **Optimized** tab.

   The Developer tool displays the optimized SQL query plan.

SQL Data Service Properties for Memory and Concurrent Requests

To optimize SQL data service performance, configure concurrency and memory properties for the Data Integration Service in the Administrator tool.

The following table describes the maximum memory per request property for the SQL Service Module:

Property	Description
Maximum Memory Per Request	The behavior of Maximum Memory Per Request depends on the following Data Integration Service configurations: - The service runs jobs in separate local or remote processes, or the service property Maximum Memory Size is 0 (default). In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to all transformations that use auto cache mode in a single request. The service allocates memory separately to transformations that have a specific cache size. The total memory used by the request can exceed the value of Maximum Memory Per Request. - The service runs jobs in the Data Integration Service process, and the service property Maximum Memory Size is greater than 0. In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to a single request. The total memory used by the request cannot exceed the value of Maximum Memory Per Request. Default is 50,000,000.

The following table describes the maximum heap size property for the Data Integration Service process:

Property	Description
Maximum Heap Size	Amount of RAM allocated to the Java Virtual Machine (JVM) that runs the Data Integration Service. Use this property to increase the performance. Append one of the following letters to the value to specify the units: - b for bytes. - k for kilobytes. - m for megabytes. - g for gigabytes. Default is 640 megabytes. **Note:** Consider increasing the heap size when the Data Integration Service needs to process large amounts of data.

The following table describes the SQL properties for the Data Integration Service process:

Property	Description
Maximum # of Concurrent Connections	Limits the number of database connections that the Data Integration Service can make for SQL data services. Default is 100.

The following table describes the execution options for the Data Integration Service:

Property	Description
Maximum On-Demand Execution Pool Size	Maximum number of on-demand jobs that can run concurrently. Jobs include data previews, profiling jobs, REST and SQL queries, web service requests, and mappings run from the Developer tool. All jobs that the Data Integration Service receives contribute to the on-demand pool size. The Data Integration Service immediately runs on-demand jobs if enough resources are available. Otherwise, the Data Integration Service rejects the job. Default is 10.
Maximum Native Batch Execution Pool Size	Maximum number of deployed jobs that can run concurrently in the native environment. The Data Integration Service moves native mapping jobs from the queue to the native job pool when enough resources are available. Default is 10.
Maximum Hadoop Batch Execution Pool Size	Maximum number of deployed jobs that can run concurrently in the Hadoop environment. The Data Integration Service moves Hadoop jobs from the queue to the Hadoop job pool when enough resources are available. Default is 100.
Maximum Memory Size	Maximum amount of memory, in bytes, that the Data Integration Service can allocate for running all requests concurrently when the service runs jobs in the Data Integration Service process. When the Data Integration Service runs jobs in separate local or remote processes, the service ignores this value. If you do not want to limit the amount of memory the Data Integration Service can allocate, set this property to 0. If the value is greater than 0, the Data Integration Service uses the property to calculate the maximum total memory allowed for running all requests concurrently. The Data Integration Service calculates the maximum total memory as follows: Maximum Memory Size + Maximum Heap Size + memory required for loading program components Default is 0. **Note:** If you run profiles or data quality mappings, set this property to 0.

Result Set Cache for an SQL Data Service

When you configure the result set cache, the Data Integration Service caches the results of the DTM process associated with each SQL data service query and web service request. The Data Integration Service caches the results for the expiration period that you configure. When a client makes the same query before the cache expires, the Data Integration Service returns the cached results.

Consider the following solution for result set cache bottlenecks:
Configure the result set cache for an SQL data service.

Result set cache enables the Data Integration Service to use cached results for SQL data service queries. Users that run identical queries in a short period of time may want to use the result set cache to decrease the runtime of identical queries.

When you enable the Data Integration Service to use cached results, data service performance increases. However, to further improve the data service processing time for identical queries, allocate enough space to store the cache in memory. When you configure the amount of cache memory to be equal to or greater than what is required to cache the results, you increase performance by reducing the system I/O overhead. When the Data Integration Service writes cache files to disk, the data service processing time increases due to system I/O overhead.

SQL Data Service Result Set Cache Properties

To increase performance, you can configure the result set cache properties for a Data Integration Service. You can also configure the number of milliseconds that the result set cache is available to use for a SQL data service.

The following table describes the result set cache properties for the Data Integration Service:

Property	Description
File Name Prefix	The prefix for the names of all result set cache files stored on disk. Default is RSCACHE.
Enable Encryption	Indicates whether result set cache files are encrypted using 128-bit AES encryption. Valid values are true or false. Default is true.

The following table describes the property that configure the number of miliseconds that the result set cache is available to the SQL data service:

Property	Description
Result Set Cache Expiration Period	The number of milliseconds that the result set cache is available for use. If set to -1, the cache never expires. If set to 0, result set caching is disabled. Changes to the expiration period do not apply to existing caches. If you want all caches to use the same expiration period, purge the result set cache after you change the expiration period. Default is 0.

Enabling Result Set Caching for an SQL Data Service

To use cached results for identical SQL data service queries, configure the Data Integration Service to use result set caching.

1. In the Administrator tool, select a Data Integration Service.

2. Click the **Process** view to configure the result set cache properties.
3. Click the **Application** view and then click the SQL data service to configure the Result Set Cache Expiration property.

Persisting Virtual Data in Temporary Tables

A temporary table is a table in a relational database that stores intermediate, temporary data. Complex queries commonly require storage for large amounts of intermediate data, such as information from joins. When you implement temporary tables, business intelligence tools can retrieve this data from the temporary table instead of the SQL data service. This results in an increase in performance.

Temporary tables also provide increased security in two ways. First, only the user of the active session can access the tables. Also, the tables persist while a session is active, and the database drops the tables when the connection closes.

Temporary Table Implementation

You can use temporary tables to improve the performance of large, complex queries. Temporary tables improve performance because queries to temporary tables on a relational database are faster than repeated queries to the SQL data service for the same data set.

Implementation of temporary tables for performance improvement requires actions by the Informatica administrator and a business intelligence tool developer.

First, the Informatica administrator creates a relational database connection, and configures the Data Integration Service to use the connection.

Then the developer for a business intelligence tool (for example, IBM Cognos or SAP Business Objects) creates a connection between the business intelligence tool and the Informatica SQL data service. The connection uses the Informatica ODBC or JDBC driver.

When these connections are active, the business intelligence tool can create and use temporary tables to process large amounts of intermediate data.

CHAPTER 9

Web Service Optimization

This chapter includes the following topics:

Web Service Optimization Overview

You can optimize web services to improve performance when the Data Integration Service runs web service requests. Tune the Data Integration Service to manage memory and handle concurrent web service requests. To improve web service performance, use web service message compression, optimize HTTP requests, and configure the data object and result set cache, and configure error log levels.

Use the following optimization techniques to optimize a web service:

- Optimize HTTP requests.
- Compress web service messages.
- Configure the web service optimizer level.
- Configure web services properties for concurrency and memory for a Data Integration process.
- Configure the Data Integration Service to keep a DTM process active so that it can process more than one web service request.
- Configure data object caching for the web service.
- Configure result set caching for the web services.
- Configure the web services run-time error log levels.

Related Topics:

- "Data Object Caching" on page 61

Optimize HTTP Requests

Optimize HTTP requests to reduce the number of requests to the web server.

Consider the following solutions for HTTP request bottlenecks:

Decrease the HTTP socket timeout for the web service client.

The socket timeout sets the amount of the time the client waits before timing out the HTTP request. The web service client can hang if the socket timeout value is large.

Web Service Message Compression

You can optimize web service performance by compressing large web messages that are passed to and from providers.

Consider the following solution for web service message bottlenecks:

Enable SOAP message compression for a web service client.

SOAP message compression enables the web service to compress web service to receive compressed web service client messages. The web service can accept a SOAP message with GZip compression from a web service client.

When the Data Integration Service receives the response from the web service, it checks the Content-Encoding HTTP header in the SOAP message and it decodes the message.

Web Service Optimizer Level

The Data Integration Service optimizes web services based on the optimizer level that you configure. Configure the optimizer level when you want the web service to use an optimizer level other than the normal. By default, each web service uses the normal optimizer level.

You can choose one of the following optimizer levels:

None

The Data Integration Service does not apply any optimization.

Minimal

The Data Integration Service applies the early projection optimization method.

Normal

The Data Integration Service applies the early projection, early selection, branch pruning, push-into, global predicate optimization, and predicate optimization methods. Normal is the default optimization level.

Full

The Data Integration Service applies the cost-based, early projection, early selection, branch pruning, predicate, push-into, semi-join, and dataship-join optimization methods.

You can use one or more of the following methods to configure the optimizer level for a web service:

- Configure the optimizer level for data preview of a web service before you deploy it to a Data Integration Service.
- Configure the optimization level for deployed web services that run on a specific Data Integration Service.
- Configure the optimizer level in the header of the web service request for a deployed web service.

Configuring the Web Service Optimizer Level for Data Preview

Configure the optimizer level that the Data Integration Services uses to preview the output of a web service.

1. In the Developer tool, click **Run** > **Open Run Dialog**.

   The **Run** dialog box appears.
2. Click **Web Service Configuration**.
3. Click the **New** button.
4. Enter a name for the web service configuration.
5. Click the **Advanced** tab.
6. Select an optimizer level.
7. Click **Apply**.
8. Click **Close**

   The Developer tool creates the web service configuration.

When you run the data viewer to preview the output of an operation mapping, select the web service configuration that includes the optimizer level that you want to use.

Configuring the Optimizer Level for Deployed Web Services

Configure the optimizer level that the Data Integration Services uses to run a deployed web service. You can choose to override the optimizer level for a single request by configuring the optimizer level in the HTTP header of the web service SOAP request.

1. In the Administrator tool, select a Data Integration Service.
2. Click the **Applications** view.
3. Expand the application that contains the web service for which you want to configure the optimizer level.

4. Select the web service and edit the following property:

Property	Description
Optimization Level	The optimizer level that the Data Integration Service applies to the object. Enter the numeric value that is associated with the optimizer level that you want to configure. You can enter one of the following numeric values: - 0. The Data Integration Service does not apply optimization. - 1. The Data Integration Service applies the early projection optimization method. - 2. The Data Integration Service applies the early projection, early selection, push-into, and predicate optimization methods. - 3. The Data Integration Service applies the cost-based, early projection, early selection, push-into, predicate, and semi-join optimization methods.

5. To override the web service optimization level for a web service request, include the following entry in the HTTP header of the web service SOAP request: `WebServiceOptions.optimizeLevel=<numeric_optimizer_level>`.

Web Services Properties for Memory and Concurrent Requests

To optimize web service performance, configure concurrency and memory properties for the Data Integration Service and each web service in the Administrator tool.

The following table describes the maximum memory per request property for the Web Service Module:

Property	Description
Maximum Memory Per Request	The behavior of Maximum Memory Per Request depends on the following Data Integration Service configurations: - The service runs jobs in separate local or remote processes, or the service property Maximum Memory Size is 0 (default). In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to all transformations that use auto cache mode in a single request. The service allocates memory separately to transformations that have a specific cache size. The total memory used by the request can exceed the value of Maximum Memory Per Request. - The service runs jobs in the Data Integration Service process, and the service property Maximum Memory Size is greater than 0. In this case, Maximum Memory Per Request is the maximum amount of memory, in bytes, that the Data Integration Service can allocate to a single request. The total memory used by the request cannot exceed the value of Maximum Memory Per Request. Default is 50,000,000.

The following table describes the execution options for the Data Integration Service:

Property	Description
Maximum Memory Size	Maximum amount of memory, in bytes, that the Data Integration Service can allocate for running all requests concurrently when the service runs jobs in the Data Integration Service process. When the Data Integration Service runs jobs in separate local or remote processes, the service ignores this value. If you do not want to limit the amount of memory the Data Integration Service can allocate, set this property to 0. If the value is greater than 0, the Data Integration Service uses the property to calculate the maximum total memory allowed for running all requests concurrently. The Data Integration Service calculates the maximum total memory as follows: Maximum Memory Size + Maximum Heap Size + memory required for loading program components Default is 0. **Note:** If you run profiles or data quality mappings, set this property to 0.

The following table describes the HTTP configuration properties for the Data Integration Service process:

Property	Description
Maximum Backlog Request	Maximum number of HTTP or HTTPS connections that can wait in a queue for this Data Integration Service process. Default is 100.
Maximum Concurrent Requests	Maximum number of HTTP or HTTPS connections that can be made to this Data Integration Service process. Minimum is 4. Default is 200. **Note:** For a web service, this property impacts the number of web service requests that the Data Integration Services accepts before it sends the requests to the Data Integration Service backlog.

The following table describes the maximum heap size property that you can configure for the Data Integration Service process:

Property	Description
Maximum Heap Size	Amount of RAM allocated to the Java Virtual Machine (JVM) that runs the Data Integration Service. Use this property to increase the performance. Append one of the following letters to the value to specify the units: - b for bytes. - k for kilobytes. - m for megabytes. - g for gigabytes. Default is 640 megabytes. **Note:** Consider increasing the heap size when the Data Integration Service needs to process large amounts of data.

Example Data Integration Service Configuration for Concurrent Web Service Requests

When you configure how the Data Integration Service processes concurrent web services requests, verify that the value for the maximum number of concurrent requests is the same for the web service and the Data Integration Service process.

For example, in the following configuration the Data Integration Service accepts 200 concurrent HTTP requests but only 10 web service concurrent requests:

Property Type	Property Name	Configuration
Data Integration Service Process	Maximum Concurrent Requests	200
Data Integration Service Process	Maximum Backlog Request	500
Data Integration Service	Maximum On-Demand Execution Pool Size	100
Web Service	Maximum Concurrent Request	10

When the Data Integration Service receives 20 web service requests, 10 web service requests fail because the web service can only accept 10 concurrent requests.

To avoid web service requests failing when the web service reaches its maximum number of concurrent requests, configure the same maximum value for the Data Integration Service process and the web service. When the number of requests sent to the Data Integration Service exceeds the maximum concurrent requests value, the additional requests remain in the backlog until the Data Integration Service process is available to process the requests.

Web Service Property to Configure an Active DTM Instance

To increase performance, you can configure the Data Integration Service to keep a DTM instance active so that it can process more than one web service request. You can configure the DTM Keep Alive Time property for the Data Integration Service in the Administrator tool.

The following table describes the DTM Keep Alive Time property:

Property	Description
DTM Keep Alive Time	Number of milliseconds that the DTM instance stays open after it completes the last request. Web service requests that are issued against the same operation can reuse the open instance. Use the keep alive time to increase performance when the time required to process the request is small compared to the initialization time for the DTM instance. If the request fails, the DTM instance terminates. Default is 5000. **Note:** The ability to use an existing DTM instance increases performance. The DIS requires additional resources to start a DTM instance for each request. Keeping the DTM active consumes memory. Therefore, users should consider the memory consumption when configuring this option.

Web Service Result Set Caching

When you configure result set caching, the Data Integration Service caches the results of the DTM process associated with each web service request. The Data Integration Service caches the results for the expiration period that you configure. When an external client makes the same request before the cache expires, the Data Integration Service returns the cached results.

Consider the following solution for result set cache bottlenecks:

Configure the result set cache for a web service.

Result set caching enables the Data Integration Service to use cached results for web service requests. Users that run identical queries in a short period of time may want to use result set caching to decrease the runtime of identical queries.

The Data Integration Service stores the result set cache for web services by user when the web service uses WSSecurity. The Data Integration Service stores the cache by the user name that is provided in the username token of the web service request. When the Data Integration Service caches the results by user, the Data Integration Service only returns cached results to the user that sent the web service request.

Enabling Result Set Caching for a Web Service

To use cached results for identical web service requests, configure the Data Integration Service to use result set caching.

1. In the Administrator tool, select a Data Integration Service.
2. Click the **Process** view to configure the result set cache properties.
3. Click the **Application** view, click the web service, and then click the operation to configure the cache expiration period in the web service operation properties. If you want the Data Integration Service to cache the results by user, enable WS-Security in the web service properties.
4. To disable result set caching for a web service request when the web service operation is configured to cache the result set, include the following syntax in the HTTP header of the SOAP request:

```
WebServiceOptions.disableResultSetCache=true
```

Web Service Log Management

System I/O performance can decrease when the Data Integration Service writes and maintains a large number of log files. The Data Integration Service generates web service run-time logs based on the trace level that you configure. Consider managing the number of log files that the Data Integration Service writes and maintains.

Consider the following solutions for web service log bottlenecks:

Set the web service trace level to off.

When you configure web service properties for a deployed web service, you can specify the log trace level. The trace level determines the types of logs that the Data Integration Service writes to the run-time log location. The default web service trace level is INFO. When the trace level is set to FINEST or ALL, performance can decrease because the Data Integration Service writes more logs to the log file. The performance impact of setting the trace level to FINEST or ALL is the greatest when the web service uses HTTPS and WS-Security.

Archive log files that are no longer required.

System I/O is affected by storing too many log files. By default, the Data Integration Services writes the web service run-time logs in the following directory: `<InformaticaInstallationDir>/tomcat/bin/disLogs/ws`

Note: If you delete the ws folder to empty the logs, you must re-create the ws folder. Stop the Data Integration Service before you delete and re-create the ws folder.

Enable the Skip Logs property in the Data Integration Service

Enable the **Skip Logs** property to prevent the Data Integration Service from generating log files when the web service request completes successfully. The tracing level for web services must be set to INFO or higher in the Data Integration Service.

CHAPTER 10

Connections Optimization

This chapter includes the following topics:

Connections Optimization Overview

You can optimize connections to improve performance. You can manage the pool of idle connection instances for a database connection. You can increase the network packet size to allow larger packets of data to cross the network at one time.

Use the following techniques to optimize connections:

- Optimize connection pooling.
- Optimize the database network packet size.

Connection Pooling

Connection pooling is a framework to cache database connection information that is used by the Data Integration Service. It increases performance through the reuse of cached connection information.

Consider the following solution for connections bottlenecks:

Enable connection pooling for a database connection.

Enable connection pooling to optimize connection performance. You can manage the idle connection instances for a database connection. The connection pool retains idle connection instances based on the pooling properties that you configure. You can adjust the maximum and minimum number of idle connections and the maximum wait time for an idle connection.

Pooling Properties in Connection Objects

You can edit connection pooling properties in the **Pooling** view for a database connection.

The number of connection pool libraries depends on the number of running Data Integration Service processes or DTM processes. Each Data Integration Service process or DTM process maintains its own connection pool library. The values of the pooling properties are for each connection pool library.

For example, if you set maximum connections to 15, then each connection pool library can have a maximum of 15 idle connections in the pool. If the Data Integration Service runs jobs in separate local processes and three DTM processes are running, then you can have a maximum of 45 idle connection instances.

To decrease the total number of idle connection instances, set the minimum number of connections to 0 and decrease the maximum idle time for each database connection.

The following list describes database connection pooling properties that you can edit in the **Pooling** view for a database connection:

Enable Connection Pooling

Enables connection pooling. When you enable connection pooling, each connection pool retains idle connection instances in memory. To delete the pools of idle connections, you must restart the Data Integration Service.

If connection pooling is disabled, the DTM process or the Data Integration Service process stops all pooling activity. The DTM process or the Data Integration Service process creates a connection instance each time it processes a job. It drops the instance when it finishes processing the job.

Default is enabled for DB2 for i5/OS, DB2 for z/OS, IBM DB2, Microsoft SQL Server, Oracle, and ODBC connections. Default is disabled for Adabas, IMS, Sequential, and VSAM connections.

Minimum # of Connections

The minimum number of idle connection instances that a pool maintains for a database connection after the maximum idle time is met. Set this value to be equal to or less than the maximum number of idle connection instances. Default is 0.

Maximum # of Connections

The maximum number of idle connection instances that a pool maintains for a database connection before the maximum idle time is met. Set this value to be more than the minimum number of idle connection instances. Default is 15.

Maximum Idle Time

The number of seconds that a connection instance that exceeds the minimum number of connection instances can remain idle before the connection pool drops it. The connection pool ignores the idle time when the connection instance does not exceed the minimum number of idle connection instances. Default is 120.

Database Network Packet Size

If you read from or write to Oracle, Sybase ASE, or Microsoft SQL Server targets, you can improve the performance by increasing the network packet size based on the database that you read from or write to. Increasing the network packet size allows larger packets of data to cross the network at one time.

Consider the following solutions for database network packet size bottlenecks:

Increase the database network packet size for an Oracle database.

You can increase the database server network packet size in listener.ora and tnsnames.ora. Consult your database documentation for additional information about increasing the packet size, if necessary.

Increase the database network packet size for a Sybase ASE database.

Consult your database documentation for information about how to increase the packet size. You must also change the packet size for Sybase ASE in the relational connection object in the Data Integration Service to reflect the database server packet size.

Increase the database network packet size for a Microsoft SQL Server database.

Consult your database documentation for information about how to increase the packet size. You must also change the packet size for Microsoft SQL Server in the relational connection object in the Data Integration Service to reflect the database server packet size.

INDEX

S

T

U

V

W