Abstract

When you use the Informatica platform to read data from or write data to relational databases, multiple factors such as hardware parameters, database parameters, and Informatica mapping parameters impact the performance. You can optimize the performance by tuning these parameters appropriately. This article describes general reference guidelines to help you tune the performance of Model repository mappings and PowerCenter repository mappings that contain relational sources or targets.

Supported Versions

- Informatica Platform 10.0

Table of Contents

Overview .............................................................. 2
Performance Tuning Areas ........................................ 3
Tune the Hardware .................................................... 3
  CPU Frequency ....................................................... 3
  NIC Card Ring Buffer Size ......................................... 3
Tune the Database ..................................................... 5
  Analyze Database Statistics to Fine Tune Queries ............. 5
  Maintain Different Physical Disks for Different Tablespaces .... 6
  Determine the Expected Database Growth ....................... 6
  Use the EXPLAIN PLAN Statement to Fine Tune Queries ....... 6
  Avoid Foreign Key Constraints .................................... 7
  Drop Indexes before Loading Data ............................... 7
  Optimize the JDBC Connection URL Parameters ............... 7
  Determine the Best Connectivity Interface ..................... 10
Tune the Mapping .................................................... 12
  Ports Precision ...................................................... 12
  Data Movement Mode ............................................. 15
  Data Type Mapping ................................................. 15
  Commit Interval .................................................... 15
  Partitioning .......................................................... 16
  Pushdown Optimization ........................................... 17
  Optimizer Hints ..................................................... 19

Overview

Performance tuning is an iterative process in which you analyze the performance, use guidelines to estimate and define parameters that impact the performance, and monitor and adjust the results as required.

This document describes the key hardware, database, and Informatica mapping parameters that you can tune to optimize the performance of the Informatica platform when the mapping contains relational sources or targets. It also includes case studies that illustrate the impact that the tuning has on the performance.
**Note:** The performance testing results listed in this article are based on observations in an internal Informatica environment using data from real-world scenarios. The performance might vary based on individual environments and other parameters even when you use the same data.

### Performance Tuning Areas

You can optimize the performance of relational database mappings by tuning the following areas:

- Hardware
- Database
- Informatica mapping

### Tune the Hardware

You can tune the following hardware parameters to optimize the performance:

- CPU frequency
- NIC card ring buffer size

#### CPU Frequency

Dynamic frequency scaling is a feature that allows the processor's frequency to be adjusted on-the-fly either for power savings or to reduce heat. Ensure that the CPU operates at least at the base frequency.

When CPUs are under clocked, that is, they run below the base frequency, the performance degrades by 30% to 40%. Informatica recommends that you work with your IT system administrator to ensure that all the nodes on the cluster are configured to run at least at their supported base frequency.

To tune the CPU frequency for Intel multi-core processors, perform the following steps:

1. Run the `lscpu` command to determine the current CPU frequency, base CPU frequency, and the maximum CPU frequency that is supported for the processor.
2. Request your system administrator to perform the following tasks:
   a. Increase the CPU frequency at least to the supported base frequency.
   b. Change the power management setting to **OS Control** at the BIOS level.
3. Run CPU-intensive tests to monitor the CPU frequency in real time and adjust the frequency for improved performance. On Red Hat operating systems, you can install a monitoring tool such as `cpupower`.
4. Work with your IT department to ensure that the CPU frequency and power management settings are persisted even in case of future system reboots.

#### NIC Card Ring Buffer Size

NIC configuration is a key factor in network performance tuning. When you deal with large volumes of data, it is crucial that you tune the Receive (RX) and Transmit (TX) ring buffer size. The ring buffers contain descriptors or pointers to the socket kernel buffers that hold the actual packet data.

You can run the `ethtool` command to determine the current configuration. For example, run the following command:

```
# ethtool -g eth0
```

The following sections show a sample output:

```
Ring parameters for eth0:
Pre-set maximums:
```
RX: 2040
RX Mini: 0
RX Jumbo: 8160
TX: 255

Current hardware settings:
RX: 255
RX Mini: 0
RX Jumbo: 0
TX: 255

The Pre-set maximums section shows the maximum values that you can set for each parameter. The Current hardware settings section shows the current configuration details.

A low buffer size leads to low latency. However, low latency comes at the cost of throughput. For greater throughputs, you must configure large buffer ring sizes for RX and TX.

Informatica recommends that you use the ethtool command to determine the current hardware settings and the maximum supported values. Then, set the values based on the maximum values that are supported for each operating system. For example, if the maximum supported value for RX is 2040, you can use the ethtool command as follows to set the RX value to 2040:

```
# ethtool -G eth0 RX 2040
```

If you set a low ring buffer size for data transfer, packets might get dropped. To find out if packets were dropped, you can use the netstat and ifconfig commands.

The following image shows a sample output of the netstat command:

![Sample Output of netstat Command]

The RX-DRP column indicates the number of packets that were dropped. Set the RX value such that no packets get dropped and the RX-DRP column shows the values as 0. You might need to test several values to optimize the performance.
The following image shows a sample output of the `ifconfig` command:

```
eth0 Link encap:Ethernet HWaddr 74:2B:CB:4F:62:9F
    inet addr:10.1.43.255 Bcast:10.1.43.255 Mask:255.255.255.0
    inet6 addr: fe80::72bb:ff:feef:623f/64 Scope:Link
    UP BROADCAST RUNNING MASTER MULTICAST MTU:1500 Metric:1
    RX packets:15923292937 errors:0 dropped:42192 overruns:0 frame:0
    TX packets:26491695749 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:1716640192541 (15.6 TIB) TX bytes:342389226682992 (31.1 TIB)
```

The status messages indicate the number of packets that were dropped.

**Tune the Database**

To optimize the performance of relational databases, perform the following tasks:

- Analyze database statistics to fine tune queries.
- Maintain different physical disks for different tablespaces.
- Determine the expected database growth.
- Use the EXPLAIN PLAN statement to fine tune queries.
- Avoid foreign key constraints.
- Drop indexes before loading data.
- Optimize the JDBC connection URL parameters.
- Determine the best connectivity interface.

**Analyze Database Statistics to Fine Tune Queries**

The most important resource for an SQL optimizer is the statistics collected for different tables within the catalog. Statistics provide information about indexes and their distribution. The SQL optimizer uses this information to decide the least expensive path that satisfies a query. Outdated or missing statistics information causes the optimizer to take a less optimized path and therefore increases the overall response time.

Have the database administrator analyze the database statistics and fine tune the queries for improved performance. For information about collecting the statistics for relational databases, see the database documentation.
Maintain Different Physical Disks for Different Tablespaces

As the size of the database increases, the hard disk I/O operations slow down. Many database providers allow users to split the database and store the data on multiple physical hard drives. Some database providers even allow users to split the contents of a table and store the data on multiple disks.

Use multiple physical disks to improve the performance. When you use multiple physical disks, I/O operations speed up significantly because more heads can fetch data in parallel.

Determine the Expected Database Growth

Indexes have a negative performance impact on DML queries. You can minimize the performance degradation by specifying an appropriate value for the fill factor when you create indexes. Set the auto extend parameter to true to ensure that the database storage size increases automatically based on the data size.

When you create an index, the data for the indexed columns is stored on the disk. When you insert new rows of data into a table or change the values in the columns, the database has to reorganize the stored data to accommodate the new rows. This reorganization can adversely impact the performance of the DML queries.

If you expect to insert rows in a table on a regular basis, specify the expected growth for an index. The parameter used for the expected growth is different for every database.

The following table lists the parameters used by different relational database management systems:

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PCTFREE</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>PCTFREE</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Fill factor</td>
</tr>
<tr>
<td>Informix</td>
<td>Fill factor</td>
</tr>
<tr>
<td>Sybase ASE</td>
<td>Fill factor</td>
</tr>
</tbody>
</table>

Use the EXPLAIN PLAN Statement to Fine Tune Queries

Most databases return the execution plan for any SELECT statement that is created by the optimizer. You can use this plan to fine tune the SQL queries and improve the performance. For example, if a mapping contains the source, Joiner transformation, Aggregator transformation, Filter transformation, and then the target, you can move the Filter transformation after the source to reduce the cost.

The following table lists the syntax for the EXPLAIN PLAN statement for relational database management systems:

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Syntax for the EXPLAIN PLAN Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>EXPLAIN PLAN FOR &lt;SQL query&gt;</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>EXPLAIN PLAN SET query no = &lt;&gt; for &lt;SQL query&gt;</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Set SHOWPLAN_ALL ON &lt;SQL query&gt;</td>
</tr>
</tbody>
</table>
Avoid Foreign Key Constraints

Foreign key constraints ensure data integrity at the cost of performance. Therefore, if performance is your primary goal, you can push the data integrity rules to the application layer.

A good example of a database design that avoids foreign key constraints is the system tables in most databases. Every major RDBMS has a set of tables known as system tables. These tables contain metadata information about user databases. Although there are relationships among these tables, there is no foreign key relationship because the database enforces the rules.

Drop Indexes before Loading Data

Consider dropping the indexes on a table before you load a large batch of data to the table. Dropping the indexes improves the performance of the INSERT statement. After you load the data, you can recreate the index again.

If you want to insert thousands of rows in a database that resides on another machine, use a temporary table to load the data. Ensure that this temporary table does not have any index. Since moving data from one table to another is much faster than loading from an external source, you can drop indexes on the primary table, move data from the temporary table to the target table, and finally recreate the indexes in the target table.

Optimize the JDBC Connection URL Parameters

If you use a JDBC connection to connect to a relational database, you can configure the following parameters in the JDBC connection URL to optimize the performance:

- BatchPerformanceWorkaround
- InsensitiveResultSetBufferSize
- LobStreamingProtocol
- BulkLoadBatchSize

The parameters that you can configure are different for each relational database. For a complete list of parameters that you can configure for relational databases and the supported values, click the following URL to see the performance tuning tips documented for each driver:

http://media.datadirect.com/download/docs/jdbc/alljdbc/index.html#page/userguide/rfi1359985946656.html#

BatchPerformanceWorkaround

The BatchPerformanceWorkaround parameter determines whether the driver must process data in batches.

In Informatica Developer, when you use a JDBC connection to write data to an Oracle or IBM DB2 database, you can improve the mapping performance by setting the BatchPerformanceWorkaround parameter to true in the JDBC connection URL. When you set the BatchPerformanceWorkaround parameter to true, the JDBC driver processes the data in batches, thereby improving the performance.
InsensitiveResultSetBufferSize

The InsensitiveResultSetBufferSize parameter determines the amount of memory that the DataDirect JDBC driver uses to cache scroll-insensitive result set data. By default, the driver caches 2048 KB of result set data in memory and writes any remaining data to the disk.

To improve the performance, you can increase the amount of result set data that the driver stores in memory and decrease the amount of data that the driver writes to the disk. The maximum cache size that you can configure is 2 GB.

You can set one of the following values:

- **-1.** The driver caches scroll-insensitive result set data in memory. If the size of the result set exceeds the available memory, an OutOfMemoryException is generated. Since the driver does not write result set data to the disk, the driver processes the data efficiently.

- **0.** The driver caches scroll-insensitive result set data in memory, up to a maximum of 2048 KB. If the size of the result set data exceeds the available memory, the driver pages the result set data to the disk. Because the result set data may be written to the disk, the driver may have to reformat the data to write it correctly to the disk.

- **<X>.** <X> denotes a positive integer that represents the size in KB of the memory buffer. The driver caches scroll-insensitive result set data in memory and uses this value to set the size of the memory buffer for caching the result set data. If the size of the result set data exceeds the available memory, the driver pages the result set data to the disk. Because the result set data may be written to the disk, the driver may have to reformat the data to write it correctly to the disk. Default is 2048 KB.

LobStreamingProtocol

The LobStreamingProtocol parameter determines whether the driver must use streaming or materialization (client caching) when it fetches LOB and XML data.

In most cases, streaming provides better performance. However, when you update LOB data by using CLOB and BLOB objects, the driver can cache the data on the client, which can reduce network round trips to the database server and improve performance.

You can set one of the following values:

- **Streaming.** The driver determines the most efficient way to return LOB and XML data based on the following conditions:
  - The database server supports streaming.
  - The application requests the data as a stream.
  
  For example, if streaming is supported by the database server and an application requests LOB data for a row as a stream, the driver uses streaming to transport the data. If streaming is not supported by the database server or if the application requests the data as a byte array or a String, the driver fully materializes the data on the client. Default is streaming.

- **Materialize.** The driver fully materializes the data on the client instead of using streaming.

BulkLoadBatchSize

The BulkLoadBatchSize parameter determines the number of rows that the driver must load to the database at a time when it bulk loads data.

You can improve the performance by increasing the number of rows the driver loads at a time so that fewer network round trips are required. Note that increasing the number of rows that are loaded also causes the driver to consume more memory on the client.

Default is 1000 rows.
Configuring JDBC Connection URL Parameters

To configure the JDBC connection URL parameters, perform the following steps:

1. Log in to Informatica Developer.
2. Click Windows > Preferences.
   - The Preferences dialog box appears.
3. Click Informatica > Connections.
4. Expand the domain name.
5. Expand the Database category, and then expand the JDBC category.
6. Select the JDBC connection for which you want to configure parameters and then click Edit.
   - The Edit Connection dialog box appears.
7. Click the Details tab.
8. Edit the JDBC connection URL to add the parameters.
   - For example, set the parameters as follows for an Oracle JDBC connection:
     jdbc:informatica:oracle://<host>:<port>;SID=<SID>;BatchPerformanceWorkaround=true;
    InsensitiveResultSetBufferSize=2048KB; LobStreamingProtocol=streaming;
     BulkLoadBatchSize=100000
   - Note: The parameters that you can configure are different for each relational database. For a complete list of parameters that you can configure for relational databases and the supported values, click the following URL to see the performance tuning tips documented for each driver:
9. Click OK to save the changes.
10. Run the mapping with the updated JDBC connection.

Case Study

When the JDBC connection URL parameters were set as follows for an Oracle mapping, the performance improved by 10x times:

jdbc:informatica:oracle://<host>:<port>;SID=<SID>;BatchPerformanceWorkaround=true;
InsensitiveResultSetBufferSize=2048KB; LobStreamingProtocol=streaming; BulkLoadBatchSize=100000
Determine the Best Connectivity Interface

The Informatica platform uses the following types of connectivity to communicate among clients, services, and other components in the domain:

- Native drivers. The Data Integration Service and PowerCenter Integration Service use native drivers to communicate with databases. Native drivers are packaged with the database server and client software. Install and configure the native database client software on the machines where the services run.

- ODBC. The ODBC drivers are installed with the Informatica services and the Informatica clients. The Data Integration Service and PowerCenter Integration Service use ODBC drivers to communicate with databases.

- JDBC. The Model Repository Service uses JDBC to connect to the Model repository database. The gateway nodes in the Informatica domain use JDBC to connect to the domain configuration repository.

When you use ODBC and JDBC drivers to write large volumes of data to a database in a record-by-record manner, the performance slows. Therefore, if you want to load data in bulk, use native drivers for improved performance.
Case Study

The following image summarizes the performance test results when the native, JDBC, and ODBC connectivity interfaces were used to read data from relational databases:
The following image summarizes the performance test results when the native, JDBC, and ODBC connectivity interfaces were used to write data to relational databases:

Tune the Mapping

You can tune the following parameters at the Informatica mapping level to optimize the performance of relational databases:

- Ports precision
- Data movement mode
- Data type mapping
- Commit interval
- Partitions
- Pushdown optimization
- Optimizer hints

Ports Precision

Precision is the maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.

You can tune the precision in Model repository mappings and PowerCenter repository mappings.

When mappings contain ports with a larger precision than required, the mapping performance degrades. Informatica recommends that you set the precision judiciously for all source ports, transformation ports, and target ports. For
instance, if a string port can handle data of a maximum of 200 characters, set the precision to 200. Do not set the precision to a high value such as 1000.

**Case Study**

Consider the following Developer tool mapping:

![Developer tool mapping diagram](image)

The mapping reads data from a flat file, applies an expression to the data, and writes the results data to a flat file. The precision for the source string ports is set to 2000 although the actual source data that is being read does not require large port sizes. The following image shows the source ports with the precision set to 2000:

**Columns**

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
<th>Format</th>
<th>Visibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_orderkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_partkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_suppkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_linenumber</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_quantity</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_extendedprice</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_discount</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_tax</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_returnflag</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_linestatus</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipdate</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_commitdate</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_receiptdate</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipinstruct</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipmode</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_comment</td>
<td>string</td>
<td>2000</td>
<td>0</td>
<td></td>
<td>Read and Write</td>
<td></td>
</tr>
</tbody>
</table>
To tune the mapping, the port sizes for the String columns were reduced to a lower value. The following image shows the source ports with the reduced precision:

<table>
<thead>
<tr>
<th>Name</th>
<th>Native Type</th>
<th>Precision</th>
<th>Scale</th>
<th>Format</th>
<th>Visibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_orderkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_partkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_suppkey</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_linenumber</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_quantity</td>
<td>double</td>
<td>15</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_extendedprice</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_discount</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_tax</td>
<td>double</td>
<td>15</td>
<td>2</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_returnflag</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_linestatus</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipdate</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_commitdate</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_receiptdate</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipinstruct</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_shipmode</td>
<td>string</td>
<td>100</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
<tr>
<td>l_comment</td>
<td>string</td>
<td>200</td>
<td>0</td>
<td>&lt;default&gt;</td>
<td>Read and Write</td>
<td></td>
</tr>
</tbody>
</table>

After tuning the precision, the mapping execution time reduced by 50%.

The following image summarizes the performance test results:
Data Movement Mode

The data movement mode specifies the mode that the Data Integration Service and PowerCenter Integration Service must use while moving data.

The data movement mode affects how the Data Integration Service and PowerCenter Integration Service enforce code page relationships and code page validation. It can also affect performance. Applications can process single-byte characters faster than multibyte characters.

You can tune the data movement mode in Model repository mappings and PowerCenter repository mappings.

Tuning Guidelines for Model Repository Mappings

When you run mappings on the Data Integration Service, the logical Data Transformation Manager (LDTM) component of the Data Integration Service determines whether to use the ASCII or Unicode data movement mode for mappings that read from a flat file or relational source. The LDTM determines the data movement mode based on the character set that the mapping processes. When a mapping processes all ASCII data, the LDTM selects the ASCII mode. In ASCII mode, the Data Integration Service uses one byte to store each character, which can optimize mapping performance. In Unicode mode, the service uses two bytes for each character.

For more information about the data movement mode, see the “Data Integration Service Architecture” chapter in the Informatica 10.0 Application Service Guide.

Tuning Guidelines for PowerCenter Repository Mappings

When you create a PowerCenter Integration Service, you can specify the mode based on the type of data you want to move, single byte or multibyte data.

For example, if the data does not contain any UTF-8 data, specify the data movement mode as ASCII.

Data Type Mapping

When the Data Integration Service or PowerCenter Integration Service reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the Data Integration Service or PowerCenter Integration Service writes data to a target, it converts the transformation data types to the comparable native data types.

You can tune the data type mapping in Model repository mappings and PowerCenter repository mappings.

When you map source ports to transformation ports and then to target ports, avoid unnecessary data type conversions. For instance, do not map a port of the string data type to a port of the date data type. Ensure that you map ports to the same data type in all components of the mapping. Also, remove all unconnected ports from the mapping.

Commit Interval

You can tune the commit interval for PowerCenter repository mappings.

The commit interval setting determines the point at which the PowerCenter Integration Service commits data to the targets. By default, the PowerCenter Integration Service uses a commit interval of 10,000 rows.

Each time the PowerCenter Integration Service commits, performance slows. Therefore, the smaller the commit interval, the more often the PowerCenter Integration Service writes to the target database, and the slower the overall performance.

If you increase the commit interval, the number of times the PowerCenter Integration Service commits decreases and performance improves.

When you increase the commit interval, consider the log file limits in the target database. If the commit interval is too high, the PowerCenter Integration Service may fill the database log file and cause the session to fail.
Therefore, weigh the benefit of increasing the commit interval against the additional time you would spend recovering a failed session.

**Note:** When bulk loading to Sybase and DB2 targets, the PowerCenter Integration Service ignores the commit interval you define in the session properties and commits data when the writer block is full.

When bulk loading to Microsoft SQL Server and Oracle targets, the PowerCenter Integration Service commits data at each commit interval. Also, Microsoft SQL Server and Oracle start a new bulk load transaction after each commit.

Therefore, when you use bulk loading for Microsoft SQL Server or Oracle targets, define a large commit interval to reduce the number of bulk load transactions and increase performance.

**Case Study**

The following image summarizes the performance test results for commit interval:

![Performance Impact of Tuning Commit Interval](image)

**Partitioning**

When you read data from or write data to relational databases, you can define partitions in order to optimize the performance of relational databases. When you configure partitioning, the Data Integration Service and PowerCenter Integration Service can perform the extract, transformation, and load tasks in parallel, thereby increasing the mapping performance. Increase the number of partitions until the performance saturation point.

**Note:** The throughput gain with increase in partitions may not always be linear. Partitioning is CPU bound. Therefore, you must configure partitioning based on the available hardware in your environment.

You can configure partitions in Model repository mappings and PowerCenter repository mappings.

The type of partitioning that is supported differs based on the type of relational database. For information on configuring partitioning for Model repository mappings, see the Informatica 10.0 Developer Mapping Guide. For information on configuring partitioning for PowerCenter repository mappings, see the PowerCenter 10.0 Advanced Workflow Guide.
Case Study

The following image summarizes the performance test results when partitioning was configured for an Oracle source:

![Performance Impact of Configuring Oracle Reader Partitioning](image)

### Pushdown Optimization

When you read data from a relational source, transform the data, and write the data target, you can configure pushdown optimization to push the mapping logic to the source or target database system. If the source and target databases are the same, you can configure full pushdown for improved performance.

When the Data Integration Service and PowerCenter Integration Service apply pushdown optimization, they push the transformation logic to the source database. The Data Integration Service and PowerCenter Integration Service translate the transformation logic into SQL queries and send the SQL queries to the database. The source database runs the SQL queries to process the transformations.

Pushdown optimization improves mapping performance when the source database can process transformation logic faster than the Data Integration Service and PowerCenter Integration Service. The Data Integration Service and PowerCenter Integration Service also read less data from the source.

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

When you configure pushdown optimization for the mapping, the Data Integration Service and PowerCenter Integration Service analyze the optimized mapping from the source to the target or until they reach a downstream transformation that they cannot push to the source database. The Data Integration Service and PowerCenter Integration Service generate and execute a SELECT statement for each source that has transformation logic pushed down. Then, the Data Integration Service and PowerCenter Integration Service read the results of this SQL query and processes the remaining transformations in the mapping.
Case Study

The following images summarize the performance test results when pushdown optimization was configured for an Oracle mapping:

### Performance Impact - PDO on Filter Tx

- **Without PDO**: 0:04:25
- **With PDO**: 0:02:55

~35% faster

### Performance Impact - PDO on Joiner Tx

- **Without PDO**: 0:12:30
- **With PDO**: 0:07:40

40% faster
Optimizer Hints

You can configure optimizer hints for Model repository mappings and PowerCenter repository mappings. If a mapping joins multiple source, you might be able to improve performance by optimizing the query with optimizing hints. Have the database administrator analyze the query, and then create optimizer hints and indexes for the source tables. 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 or PowerCenter 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 or PowerCenter 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.
Authors

Siddiq Hussain
Senior QA Engineer

Anu Chandrasekharan
Lead Technical Writer