# HP Vertica Integration with Pentaho Data Integration (PDI) Tips and Techniques

HP Vertica Analytic Database

HP Big Data Foundations
Document Release Date: February, 2015

# Legal Notices

## Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

## Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

## Copyright Notice



## Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation. UNIX® is a registered trademark of The Open Group.

# Contents

# About HP Vertica Tips and Techniques

HP Vertica develops Best Practices documents to provide you with the information you need to use HP Vertica with third-party products. This document provides guidance using one specific version of HP Vertica and one version of the vendor's software. While other combinations are likely to work, Hewlett-Packard may not have tested the specific versions you are using.

# Overview

This document provides guidance for configuring Pentaho Data Integration (PDI, also known as Kettle) to connect to HP Vertica. This document covers only PDI. However, connectivity options for other Pentaho products should be similar to the options this document provides.

The content in this document has been tested for PDI 5.1.0 and HP Vertica 7.0. Most of the information will also apply to earlier versions of both products.

# Connecting HP Vertica and Pentaho (PDI)

Pentaho Data Integration (PDI) supports both ODBC and JDBC for connectivity. Hewlett Packard recommends using JDBC, because ODBC is usually slower than JDBC, and Pentaho does not support ODBC for subscription customers.

PDI provides two ways to connect to a database via JDBC. Both drivers ship with the PDI software:

- HP Vertica-specific JDBC driver

  Hewlett Packard recommends that you use the HP Vertica-specific JDBC connector for your ETL jobs. When creating a new connection to HP Vertica, make sure you select the connector that matches your database.

- Generic JDBC driver

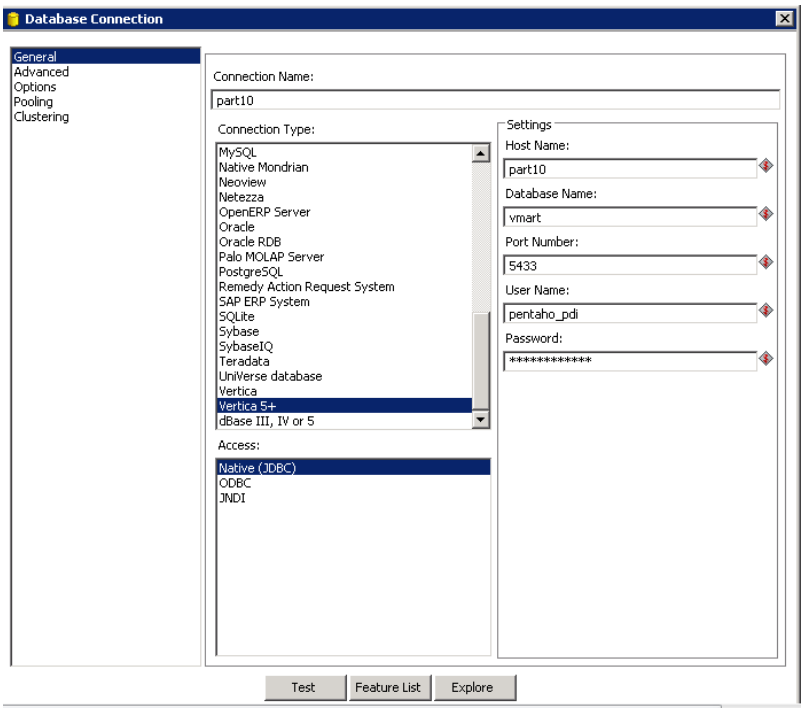
The database JDBC files are located in the following folders of your PDI installation:

- For client installations (Spoon, Pan, Kitchen, Carte): `data-integration/lib`
- For server installations: `data-integration-server/tomcat/lib`

If the client is installed on the same machine as the server, you must copy the HP Vertica JDBC jar file to both of these folders.

## Installing Your JDBC Driver

Store your HP Vertica JDBC driver file in a folder similar to `C:\<install_dir>\data-integration\lib\` (for a Windows system). The following screenshot shows how to connect to Vertica.




## HP Vertica Client Driver/Server Compatibility

Usually, each version of the HP Vertica server is compatible with the previous version of the client drivers. This compatibility lets you upgrade your HP Vertica server without having to immediately upgrade your client software. However, some new features of the new server version may not be available through the old drivers.

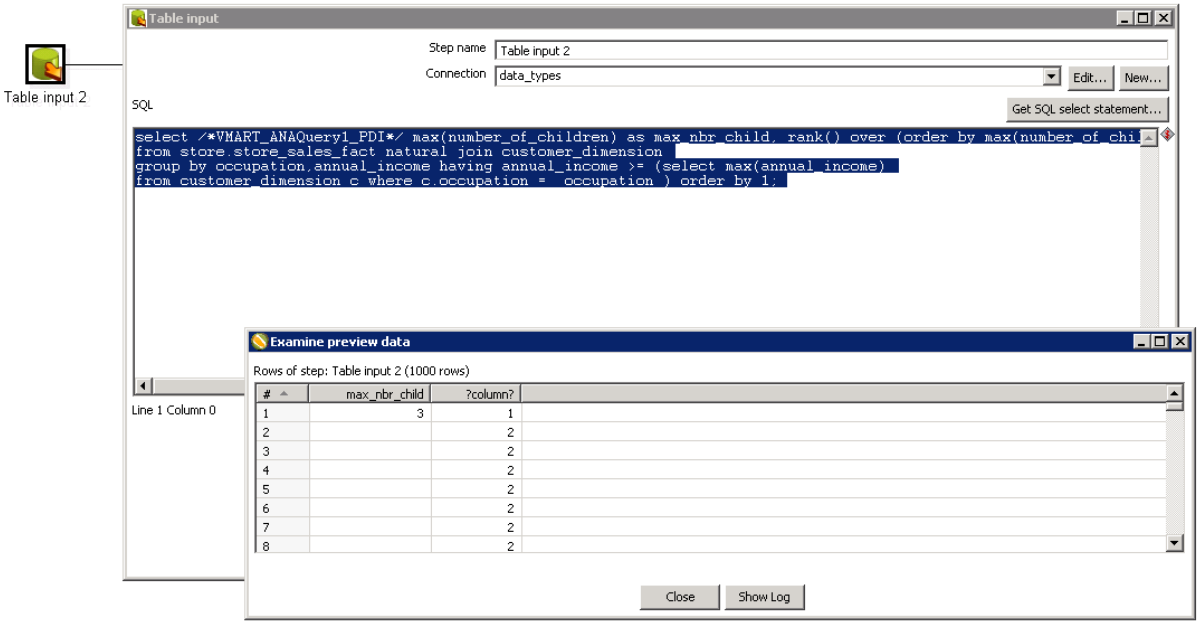
The following table summarizes the compatibility of each recent version of the client drivers with the HP Vertica server versions.

| HP Vertica Client Driver Version | Compatible HP Vertica Server Versions |
|---|---|
| 6.1.x | 6.1.x, 7.0.x, 7.1.x |
| 7.0.x | 7.0.x, 7.1.x |
| 7.1.x | 7.1.x |

Only the first two digits in the version number matter for client/server compatibility. For example, a 7.0.x client driver can talk to any 7.0.x server. For more information about client driver/server compatibility, see the Connecting to Vertica guide.

## Reading Data from HP Vertica with Input Components

PDI provides an input component that you can use to run SELECT statements on the HP Vertica database, as the following figure shows:




# Writing Data to HP Vertica

You can write data to an HP Vertica table using either of two PDI components:

- Standard table output component
- HP Vertica Bulk Loader for PDI

In limited testing, the HP Vertica Bulk loader is faster than using the standard table output component. However, test the integration on your own system to verify that all necessary capabilities are available and that your service-level agreements can be met.

## Using HP Vertica COPY

The COPY statement bulk loads data into an HP Vertica database. You can load one or more files, or pipes on a cluster host. You can also load directly from a client system, using the COPY statement with its FROM LOCAL option.

Raw input data must be in UTF-8, delimited text format. Data is compressed and encoded for efficient storage. For more information about the COPY command, see COPY in the product documentation.

When trying to decide which HP Vertica component to use, there are three basic type of ETL flows to consider:

| Type of Load | Use this COPY Option | Results |
|---|---|---|
| Small bulk load COPY (< 100 MB) | AUTO | • Writers to WOS<br>• Spills to ROS when WOS overflows |
| Large bulk load COPY | DIRECT | • Writes to WOSE<br>• Each commit becomes a new ROS container |
| Incremental load COPY | TRICKLE | • Writes to WOS<br>• Errors when WOS overflows |

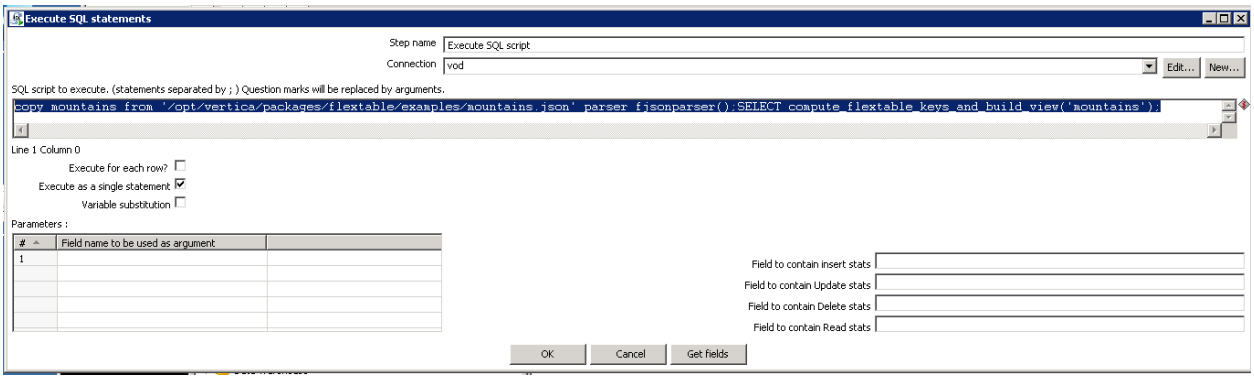
### Bulk Loading Guidelines

- When bulk loading, use COPY when possible. Using COPY
    - Reduces overhead of per-plan INSERT cost

- Avoids fragmenting the WOS

- Use INSERT when the row count is small (< 1000 rows)
- Load multiple streams on different nodes

**HP Vertica COPY Example: Execute SQL Script Statement Component**
You can load data directly into HP Vertica from a client system using the COPY statement with the FROM LOCAL option. PDI components generate only the COPY LOCAL option. However, if your data files are on the HP Vertica cluster, you can use COPY with the PDI Execute SQL Script statement component. The following example shows how to use this component to load an HP Vertica flex table:



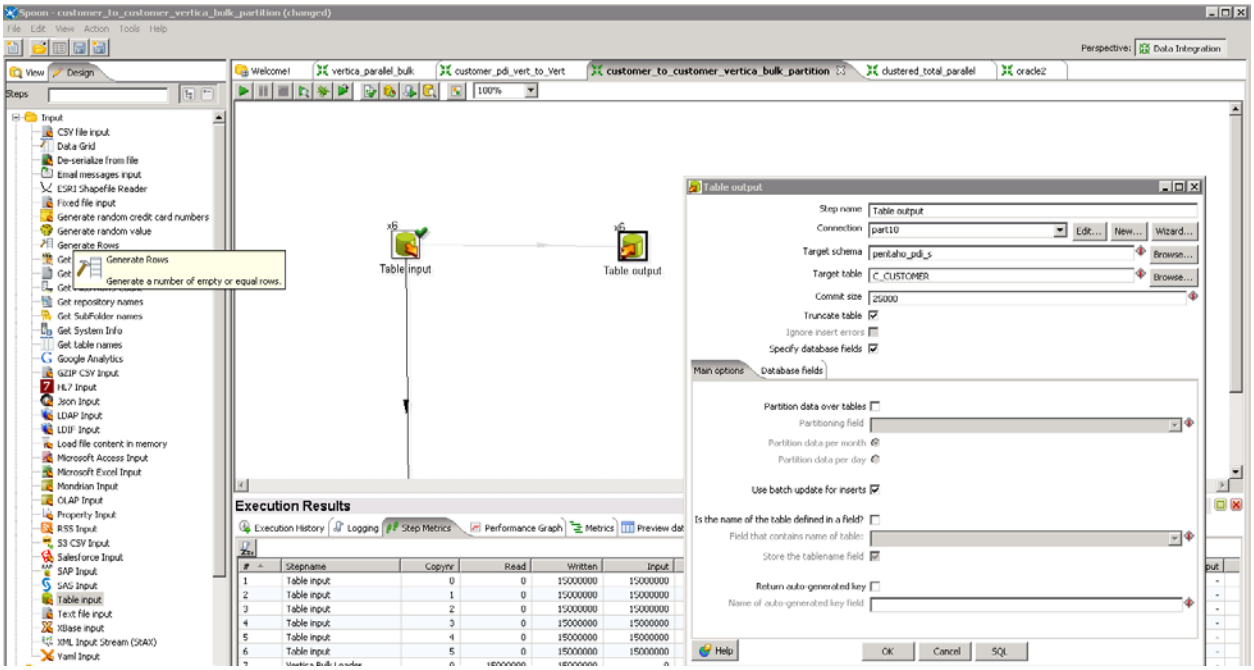

You can enter any valid SQL such as:

```
=> COPY mountains FROM
   '/opt/vertica/packages/flextable/examples/mountains.json'
   PARSER fjsonparser();
=> SELECT compute_flextable_keys_and_build_view('mountains');
```

By default, COPY uses the DELIMITER parser to load raw data into the database. Raw input data must be in UTF-8, delimited text format. Data is compressed and encoded for efficient storage. If your raw data does not consist primarily of delimited text, specify the parser that COPY should use to align most closely with the load data.

## Loading Data into HP Vertica with the Standard Table Output Component

The following figure shows the standard table output component:




In this example, the standard table output component generates the following COPY statement:

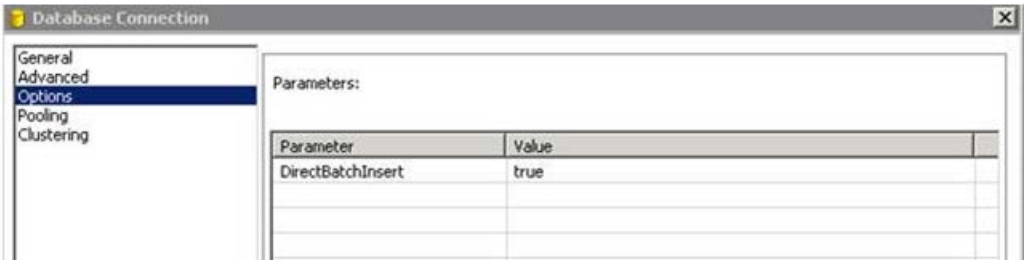
```
COPY pentaho_pdi_s.C_CUSTOMER ( C_CUSTKEY, C_NAME, C_ADDRESS, C_NATIONKEY,
C_PHONE, C_ACCTBAL, C_MKTSEGMENT, C_COMMENT ) FROM LOCAL STDIN NATIVE VARCHAR
ENFORCELENGTH RETURNREJECTED AUTO NO COMMIT
```

The DIRECT clause is *not* included in this SQL statement. This example writes the data to the HP Vertica Write Optimized Store (WOS), which is *not* optimized for large data loads. By default, the Standard Table Output component writes data to the WOS.

For a large data load, load the data directly to the HP Vertica Read Optimized Store (ROS), as described in the next section.

### Writing Data Directly to the ROS

To load data into the ROS, use the DIRECT keyword on the COPY statement. To make this happen, when using the Standard Output Table component, add the DirectBatchInsert parameter to the connection and set it to true, as in the following figure.

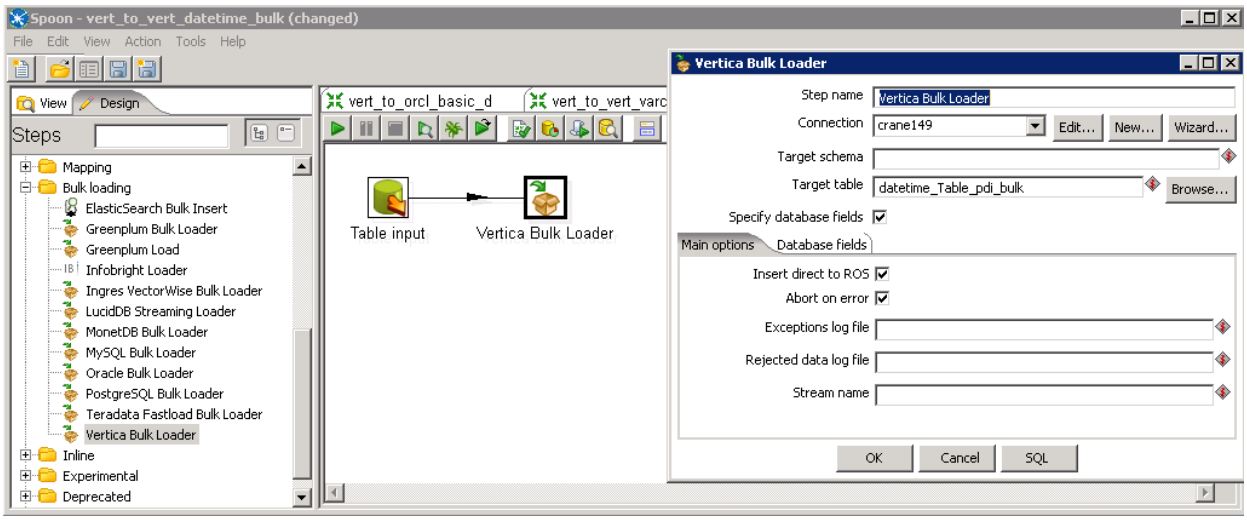



Adding the DirectBatchInsert parameter generates the following SQL:

```
COPY pentaho_pdi_s.C_CUSTOMER ( C_CUSTKEY, C_NAME, C_ADDRESS, C_NATIONKEY, C_PHONE,
C_ACCTBAL, C_MKTSEGMENT, C_COMMENT ) FROM LOCAL STDIN NATIVE VARCHAR ENFORCELENGTH
RETURNREJECTED DIRECT NO COMMIT
```

## HP Vertica Bulk Loader for PDI

PDI provides a HP Vertica Bulk Loader for PDI plugin, as shown:




By default, the HP Vertica Bulk Loader for PDI plugin is included in the PDI enterprise version. If you are using the community edition, you can download the plugin for the PDI community edition at Project Kettle-VerticaBulkLoader.

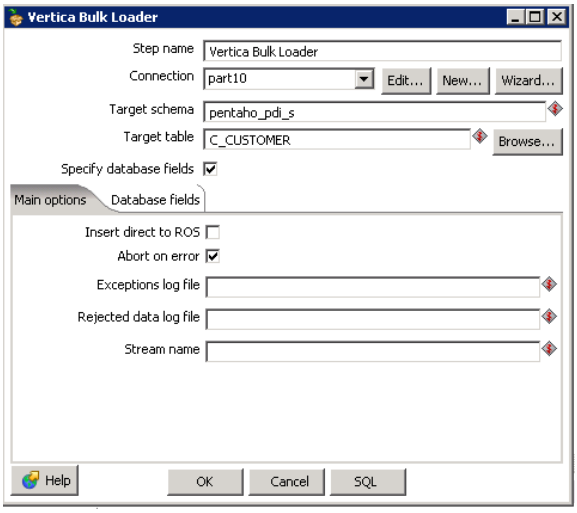
### Bulk Loading Using vsql and Named Pipes

If you are working in a Linux environment, to leverage the HP Vertica vsql client, try the universal bulk loading approach for native bulk loading and PDI, as described in Implementing Universal Bulk-Loading in PDI.

This approach is only available on Linux systems.

### Writing Data to the WOS

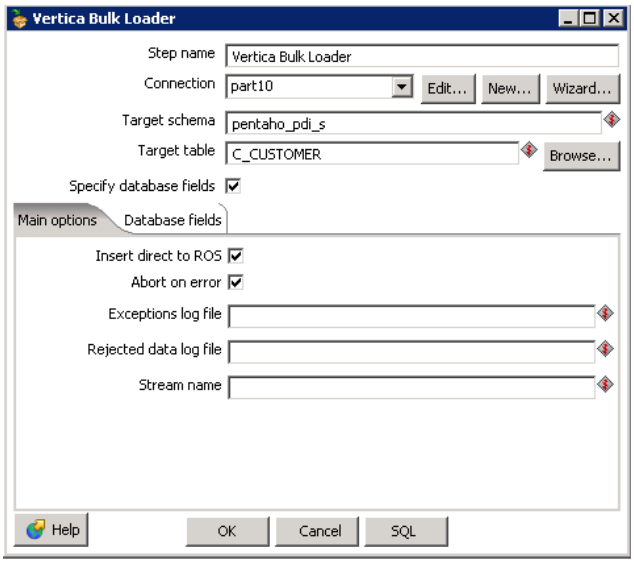
When using the HP Vertica Bulk Loader to write the data to the WOS, verify that the **Insert direct to ROS** checkbox is *not* selected:




### Writing Data Directly to the ROS

To specify that the HP Vertica Bulk Loader write data directly to the ROS, select **Insert direct to ROS**. If using INSERT INTO SELECT FROM syntax in ETL transform with large volumes, be sure to use the following syntax:

```
INSERT /+*direct*/ INTO table SELECT…
```




## Monitoring Data Load

You can monitor the progress of in-progress data loads using either standard SQL or by using the HP Vertica Management Console.

### Using SQL to Monitor Data Loads

Use the following SQL statements to monitor the progress of your data load:

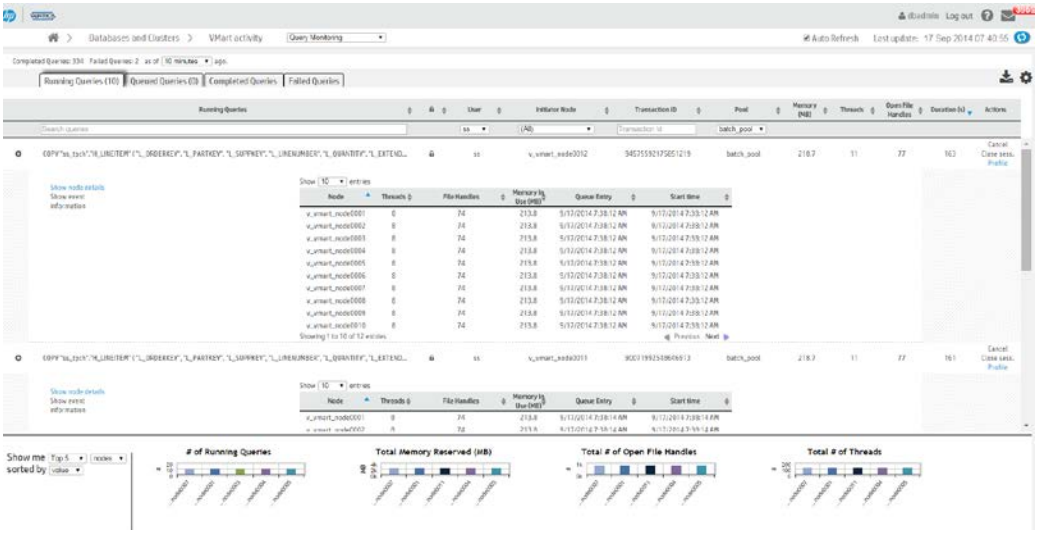
```
\echo ..Load streams
SELECT
table_name AS table,
stream_name AS stream,
TO_CHAR(DATE_TRUNC('second', current_timestamp - load_start::TIMESTAMP),
'hh24:mi:ss') AS run_time,
TO_CHAR(load_start::TIMESTAMP, 'yyyy-mm-dd hh24:mi') AS load_start,
TO_CHAR(accepted_row_count, '999,999,999,999') AS accepted,
TO_CHAR(rejected_row_count, '999,999,999,999') AS rejected,
TO_CHAR(unsorted_row_count, '999,999,999,999') AS unsorted,
TO_CHAR(sorted_row_count, '999,999,999,999') AS sorted,
sort_complete_percent AS sort_pct,
TO_CHAR(accepted_row_count/extract(EPOCH FROM current_timestamp -
load_start::TIMESTAMP), '999,999') AS rps,
TO_CHAR(sorted_row_count/extract(EPOCH FROM current_timestamp -
load_start::TIMESTAMP), '999,999') AS sort_rps
FROM load_streams ls;

\echo ..Load Totals
SELECT
LPAD(COUNT(DISTINCT table_name)::CHAR, MAX(LENGTH(table_name))) AS tables,
LPAD(COUNT(DISTINCT stream_name)::CHAR, MAX(LENGTH(stream_name))) AS streams,
TO_CHAR(DATE_TRUNC('second', current_timestamp - MIN(load_start::TIMESTAMP)),
'hh24:mi:ss') AS run_time,
TO_CHAR(MIN(load_start), 'yyyy-mm-dd hh24:mi') AS load_start,
TO_CHAR(SUM(accepted_row_count), '999,999,999,999') AS accepted,
TO_CHAR(SUM(rejected_row_count), '999,999,999,999') AS rejected,
TO_CHAR(SUM(unsorted_row_count), '999,999,999,999') AS unsorted,
```

```
TO_CHAR(SUM(sorted_row_count), '999,999,999,999') AS sorted,
SUM(sort_complete_percent) AS sort_pct,
TO_CHAR(SUM(accepted_row_count)/EXTRACT(EPOCH FROM current_timestamp -
MIN(load_start::TIMESTAMP)), '999,999') AS rps
FROM load_streams ls;
```

**Using Management Console to Monitor Data Loads**

Use the HP Vertica Management Console to better understand resource usage when jobs are executing. The following figure shows various running COPY statements and the resources consumed by the ETL job. Below the list of statements, sortable charts display the resources consumed by the batch load job that is running.




**Downloading and Installing Management Console**

Download the appropriate version of Management Console from http://my.vertica.com. Follow the instructions in the HP Vertica Installation Guide.

After the installation, you can access Management Console at https://xx.xx.xx.xx:5450/, where xx.xx.xx.xx is the IP address or host name where the Management Console is installed.

For more information about Management Console:

- Installation
- Management Console Overview
- Management Console videos:
  - http://www.vertica.com/customer-experience/vertica-101/managment-console/
  - http://www.vertica.com/customer-experience/vertica-101/managment-console/
  - http://www.vertica.com/customer-experience/vertica-101/managment-console/

## Validating Load Results

HP Vertica does not enforce constraints at data load. Constraints are enforced when data is loaded into a table with a pre-join dimension, or when the table is joined to a dimension table during a query. Therefore, you may experience constraint errors in any of these scenarios:

- If there is not exactly one dimension row that matches each foreign key value.
- An inner join query is processed. An outer join is treated as an inner join due to the presence of foreign key.
- A new pre-join projection anchored on the table with the foreign key constraint is refreshed.

- If your dimension tables have duplicate values in their primary key columns or do not have foreign keys in the fact tables. HP Vertica may return errors indicating constraint violations, because *HP Vertica checks these constraints only as part of a join and does NOT do referential integrity (RI) check at load time.*

Note: If you are testing your queries for performance, you may not see the full query speed right away. Delays in performance can occur because HP Vertica can be busy reorganizing the data for several hours. The amount of time that data reorganization requires depends on how much data was loaded and the type of load.

## Best Practices for Loading and Updating Data

To avoid creating too many or too few ROS containers, follow these recommendations:

- Load and delete data less frequently in larger batches**.**
- If you are working with many small files of raw data, use a single COPY statement to load these files at one time. Use the wildcard support to COPY from all files, or concatenate all files to a named pipe and have COPY load from the pipe.
- Use a single DELETE/UPDATE statement to delete or update batches of rows**.** When possible, delete or update batches of rows at a time to avoid creating too many delete vector ROS containers.

There are many possible reasons as to why a load is not performing to expectations. Both Pentaho and HP Vertica provide a number of documents and best practices that can help you troubleshoot data load issues.

### PDI Best Practices for Data Load

Hewlett Packard endorses the data load recommendations included in two presentations given at Pentaho World 2014:

- "Best Practices from the Trenches: How to Best Monitor and Tune PDI Transformations to Maximize Performance"
- "PDI Best Practices"

Review these recommendations and other materials that describe how to isolate bottlenecks in ETL jobs. Some recommendations from these materials include:

- Use Pentaho Operations Mart to capture metrics on job.
- Turn on Performance Monitoring and Logging.
- Create a high-level flow for investigating the root cause.
- Analyze the performance of the transformation.
- Review the transformation.
- Review the job workflow.

The recommendations to follow vary depending on execution speed and memory consumption. Be sure to set the threshold for the number of rows that can wait to be processed by the following step. If the number of waiting rows is reached, the source step waits for bandwidth to process the waiting steps. When there is bandwidth, more rows are put into the output buffer. For details, see Using Row Buffers to Troubleshooting PDI in this document.

Note: For more information about PDI transformations and steps, see Transformations, Steps, and Hops in the PDI documentation.

Consider the following root causes for transformation flow problems:

- Lookups
- Scripting steps
- Memory hogs
- Lazy conversion
- Blocking step
- Commit size, Rowset size

Consider the following root causes for job workflow problems:

- Operating system constraints—memory, network, CPU
- Parallelism

- Looping
- Execution environment

For more information about optimizing data loads with PDI:

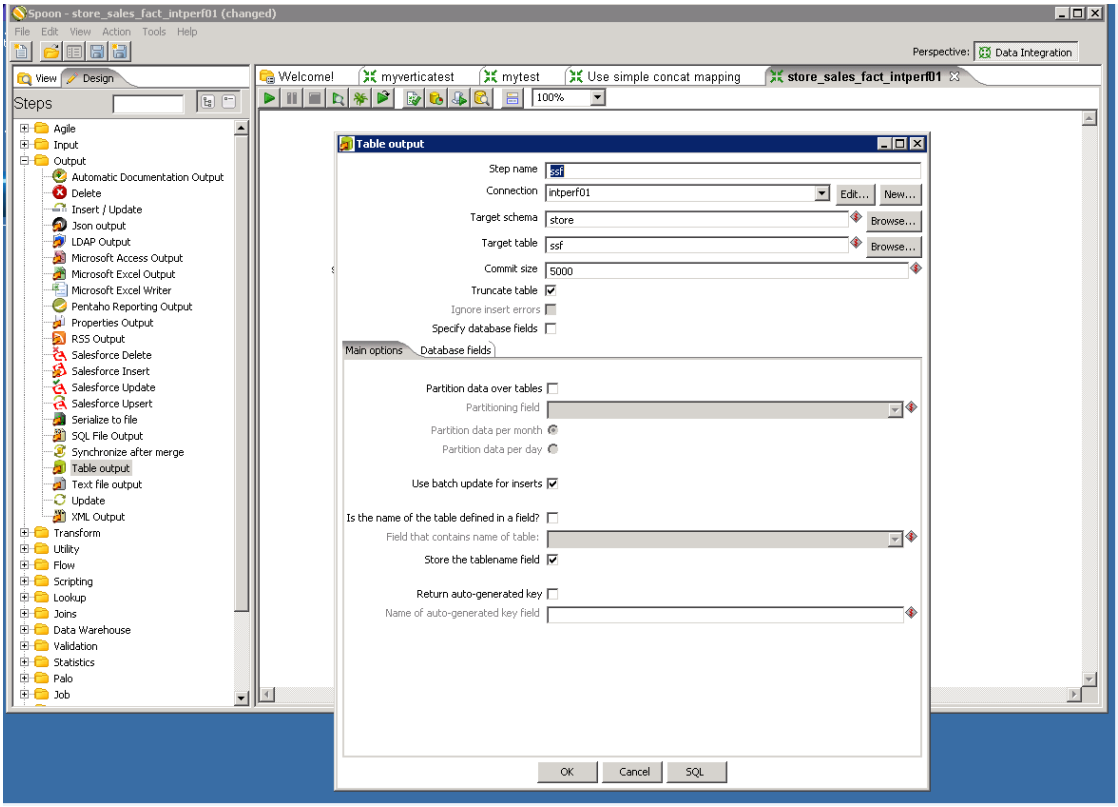
- There is a fantastic book on optimizing PDI called *Pentaho Kettle Solutions*. This book describes advanced operations like clustering PDI servers.
- Pentaho blog: http://blog.pentaho.com/author/mattcasters/
- Pentaho Community Wiki: http://wiki.pentaho.com/display/COM/Community+Wiki+Home

**Improving the Performance of Data Load**

At a minimum, consider two PDI settings to improve the performance for a particular load.

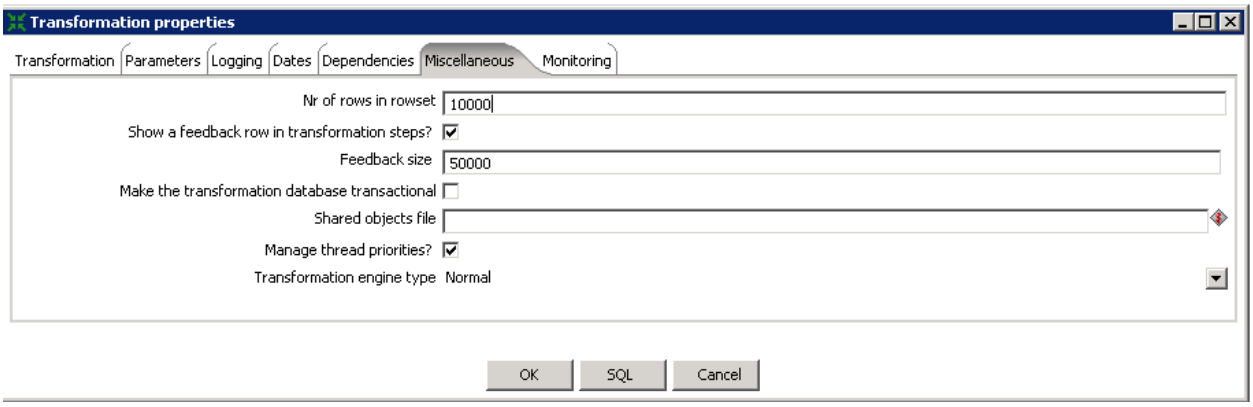
### COMMIT Size

The size of a COMMIT operation is important for data load performance, Based on testing, 25000 rows seems to be a good place to start. You may have to modify the commit size depending on the source system table size and number of columns.




### Number of Rows in Row Set

The number of rows in the RowSet setting depends on the machine memory and row set size. To set this option, go to the **Transformation properties** dialog box, the **Miscellaneous** tab. For detailed recommendations, see Performance Tuning.




# Configuring HP Vertica to Work with PDI

Consider these tips when using HP Vertica with PDI:

- Managing HP Vertica Resources
- Managing HP Vertica Storage Containers
- Managing HP Vertica ROS Containers
- Managing HP Vertica WOS Containers
- Using Linux to Monitor Resource Usage
- Exporting Data from the Source Database to Flat Files
- Data Type Mappings
- Loading Data over the Network
- Using Non-ASCII Data
- Internationalization

## Managing HP Vertica Resources

To prevent query users from being affected by ETL jobs, create a separate resource pool for PDI. Your resource pool settings depend on the amount of memory on the machine and the number of resource pools available for users. The following SQL statements show how to create a PDI resource pool:

```
=> CREATE RESOURCE POOL batch_pool MEMORYSIZE '4G' MAXMEMORYSIZE '84G'
   MAXCONCURRENCY 36;
=> DROP USER pentaho_pdi;
=> DROP SCHEMA pentaho_pdi_s;
=> CREATE SCHEMA pentaho_pdi_s;
=> CREATE user pentaho_pdi identified by 'pentaho_pwd' SEARCH_PATH
   pentaho_pdi_s;
=> GRANT USAGE on SCHEMA pentaho_pdi_s to pentaho_pdi;
=> GRANT USAGE on SCHEMA PUBLIC to pentaho_pdi;
=> GRANT USAGE on SCHEMA online_sales to pentaho_pdi;
=> GRANT USAGE on SCHEMA store to pentaho_pdi;
=> GRANT SELECT on ALL TABLES IN SCHEMA PUBLIC to pentaho_pdi;
=> GRANT SELECT on ALL TABLES IN SCHEMA store to pentaho_pdi;
=> GRANT SELECT on ALL TABLES IN SCHEMA online_sales to pentaho_pdi;
=> GRANT ALL PRIVILEGES ON SCHEMA pentaho_pdi_s TO pentaho_pdi WITH
   GRANT OPTION;
=> GRANT CREATE ON SCHEMA pentaho_pdi_s to pentaho_pdi;
=> GRANT USAGE ON RESOURCE POOL batch_pool to pentaho_pdi;
=> ALTER USER pentaho_pdi RESOURCE POOL batch_pool;
```

## Managing HP Vertica Storage Containers

HP Vertica supports INSERT, UPDATE, DELETE, and bulk load operations (COPY). You can intermix these operations with queries in a typical data warehouse workload. The storage model consists of three components that operate identically on each HP Vertica node:

| Component | Description |
|---|---|
| Write-Optimized Store (WOS) | A memory-resident data structure for storing INSERT, UPDATE, DELETE, and COPY (without /*+DIRECT*/ hints) actions. To support very fast data load speeds, the WOS stores records without data compression or indexing. The WOS organizes data by epoch and holds both committed and uncommitted transaction data. |
| Read Optimized Store (ROS) | A highly optimized, read-oriented, disk storage structure. The ROS makes heavy use of compression and indexing. You can use the COPY...DIRECT and INSERT (with /*+DIRECT*/ hints) statements to load data directly into the ROS. |
| Tuple Mover (TM) | The database optimizer component that moves data from memory (WOS) to disk (ROS). The Tuple Mover runs in the background, performing some tasks automatically at time intervals determined by its configuration parameters. |

For more information on HP Vertica ROS and WOS, see Loading Data into the Database, in HP Vertica Best Practices for OEM Customers.

## Managing HP Vertica ROS Containers

HP Vertica creates ROS containers:

- With every moveout
- Each time COPY DIRECT is executed
- Whenever a table is partitioned

ROS containers consume system resources, so the number of ROS containers has a direct effect on system performance. A large number of ROS containers can degrade performance. Too few ROS containers can prevent the system from taking advantage of inherent parallelism. Too many ROS containers can also result in HP Vertica errors.

For best performance, Hewlett-Packard recommends:

- When not loading data, 10 containers per projection
- During data load, up to 20 containers per projection
- No more than 50 containers per projection

### Too Many ROS Containers Error

ROS container exceptions can occur if you accidently target frequent, small loads directly to ROS instead of WOS. Too many ROS containers can exist if the batch load API does not correctly convert INSERT commands into COPY statements. If this error occurs, HP Vertica records the following message in the log:

```
ERROR:  Too many ROS containers exist for the following projections:
customer_100061.ark_DBD_1_seg_ptoole2_ptoole2 (limit = 1,000,000, ROS files =
999673, DV files = 0, new files = 479)
HINT:  Please wait for the tuple mover to catch up. Use 'select * from
v_monitor.tuple_mover_operations;' to monitor.
```

## Managing HP Vertica WOS Containers

The WOS can use a maximum of 25% of the physical memory on each node. HP Vertica issues a WOS overflow error when you reach or exceed this limit. For example, suppose you have up to 1 GB available for your WOS. If you fill the 1 GB before the Tuple Mover's Moveout operation can process the data, a WOS Overflow error occurs.

## Using Linux to Monitor Resource Usage

Use the following Linux commands to monitor disk and resource usage:

- `df`—Use `df` to monitor the disk so that enough space remains available.

- `vmstat`—Use `vmstat` to report monitor CPU usage at specified intervals. The column values bi (blocks in) and bo (blocks out), are reported in kilobytes per second.

  The `vmstat` display also shows swapping I/O using columns si (swap in) and so (swap out), also in kilobytes per second.

  When you use `vmstat`, you can estimate the disk bandwidth used in the load in kilobytes per second. The maximum values observed for the bi + bo columns provide the disk bandwidth estimate.

  For example, suppose you run `vmstat 60` to obtain a report every 60 seconds. During the load, you see high CPU usage in bursts for the sequence of memory sorts in the first part of chunk processing. During the rest of the process, you observe high block I/O (`vmstat` columns bi and bo). The system should not become idle until the load is done, where idle means low both CPU use and low bi + bo.

  The vmstat display also shows swapping I/O using columns si and so, for swap in and swap out, also in KB/sec. Suppose the swapping I/O is significant (si + so is more than 20% of maximum-seen bi + bo, thus stealing up to 20% of the disk bandwidth) over many minutes, especially during the high block I/O periods. In this situation, the system is under stress. The parallelism of the load should be reduced at the earliest opportunity, by reducing the number of load streams after the COPY for a chunk is finished.

  `sar` and `sar -r` can also be used.

## Exporting Data from the Source Database to Flat Files

If, for any reason, you cannot connect directly to the source database, you need to export data from the source system to flat files. Before doing so, be aware of these considerations:

- Smaller tables generally fit into a single load file. Split any large tables into 250–500 GB load files. For example, a 10 TB fact table requires 20–40 load files to maintain performance.
- The default delimiter for the COPY statement is a vertical bar (`|`). Before loading your data, verify that no CHAR(N) or VARCHAR(N) data values include this delimiter character. To test for the existence of a specific character in a column, use a query similar to the following:

  ```
  => SELECT COUNT(*) FROM t WHERE x LIKE '%|%'
  ```

  If only a few rows contain `|`, you can eliminate them from the load file using a WHERE clause. Then, load the rows separately using a different delimiter.

## Data Type Mappings

PDI has the ability to generate HP Vertica DDL automatically when moving data from another database to an HP Vertica database. Make sure that you understand some of the differences between the two databases. For example, Oracle supports NVARCHAR and HP Vertica does not. If the source system contains multi-byte character data, when converting the DDL from the source to the HP Vertica target, you must increase the amount of characters for a VARCHAR or CHAR column by at least a factor of 3. Doing so allows for the possibility of non-UTF 8 character sets. For additional information, see Locales in this document.

Contact HP Vertica Customer Support for example scripts on how to convert DDL from SQL Server and Oracle to HP Vertica.

### HP Vertica and Oracle Data Type Mappings

Oracle uses proprietary data types for all common data types (for example, VARCHAR, INTEGER, FLOAT, and DATE). If you plan to migrate your database from Oracle to HP Vertica, Hewlett Packard strongly recommends that you convert the schema. Doing so is a simple and important exercise that can minimize errors and the time lost spent fixing erroneous data issues.

For a table comparing Oracle and HP Vertica data types, see Data Type Mappings for HP Vertica and Oracle databases in the product documentation.

For information about the SQL data types that HP Vertica supports, see SQL Data Types in the product documentation. For example, if you are migrating from Oracle, you must convert the non-standard type named NUMBER to SQL-standard INT or INTEGER. The required foreign key clauses can be in the table definitions themselves or in separate ALTER TABLE t ADD CONSTRAINT ... commands. The foreign key constraints are very important to guide the Database Designer in its work.

### Expected Results for Mapping Exceptions

Depending on the type of SQL insert generated by PID, mapping exceptions can have varying results:

| Mapping Exceptions | HP Vertica COPY LOCAL | SQL INSERT |
|---|---|---|
| CHAR field with *n* characters mapped to CHAR column with less than *n* characters | Record is inserted with silent truncation. | Record is not inserted; warning message is issued. |
| EN (Edited Numeric) field mapped to INTEGER column with data that forces numeric overflow | Record is rejected and sent to the rejected records log file. | 0 is silently loaded in place of the number that would cause the overflow. |
| EN field mapped to NUMERIC column with data that exceeds the scale | Record is rejected and sent to the rejected records log file. | Record with the value that exceeds the scale is silently rejected. |

## Loading Data over the Network

A 1 Gbps (gigabits per second) network can deliver about 50 MB per second or 180 GB per hour. HP Vertica can load about 30–50 GB per hour/node for a K-safety = 1 projection design. Therefore, you should use a dedicated 1 Gbps LAN. Using a LAN with a performance that is less than 1 Gbps is proportionally slower. Hewlett-Packard recommends not loading data across an external network to avoid delays. Such delays over distance slow down the TCP protocol to a small fraction of its available bandwidth, even without competing traffic.

Note: The actual load rates that you obtain can be higher or lower. These rates depend on the properties of the data, the number of columns, the number of projections, and hardware and network speeds. You can improve load speeds further by using multiple parallel load streams.

## Using Non-ASCII data

HP Vertica stores data in the UTF-8 compressed encoding of Unicode. The resulting UTF-8 codes are identical to ASCII codes for the ASCII characters (codes 0–127 in one byte). Because all current operating systems treat ASCII in the same way, table data (CHAR columns) in all ASCII transports are also stored in UTF-8 compressed format. For UTF-8 data, verify that the extraction method you use does not convert CHAR column values to the current (non-UTF-8) locale of the source system.

On most UNIX systems, use the `locale` command to see the current locale. You can change the locale for a session by setting the LANG environment variable to en_US.UTF-8.

Sometimes, data actively uses the non-ASCII characters of Latin-1, such as the Euro sign (€) and the diacritical marks of many European languages. If you have data in another character encoding such as Latin-1 (ISO 8859), you must convert it to UTF-8. To perform the conversion, use the Linux tool `iconv`. Large fact tables are unlikely to contain non-ASCII characters, so these conversions are usually needed only for data in tables with smaller dimensions.

## Internationalization

HP Vertica supports the following internationalization features, describes in the sections that follow:

- Unicode character encoding
- Locales

For more information on configuring internationalization for your database, see Internationalization Parameters in the product documentation.

### Unicode Character Encoding: UTF-8 (8-bit UCS/Unicode Transformation Format)

All input data received by the database server must be in UTF-8 format. All data output by HP Vertica must also be in UTF-8 format. The ODBC API operates on data in:

- UCS-2 on Windows systems
- UTF-8 on Linux systems

A UTF-16 ODBC driver is available for use with the DataDirect ODBC manager.

JDBC and ADO.NET APIs operate on data in UTF-16. The HP Vertica client drivers automatically convert data to and from UTF-8 when sending to and receiving data from HP Vertica using API calls. The drivers do not transform data that you load by executing a COPY or COPY LOCAL statement.

**Locales**

The locale parameter defines the user's language, country, and any special variant preferences, such as collation. HP Vertica uses the locale to determine the behavior of various string functions. It also uses the locale when collating various SQL commands that require ordering and comparison. Such commands can include, for example, GROUP BY, ORDER BY, joins, and the analytic ORDER BY clause.

By default, the locale for the database is `en_US@collation=binary` (English US). You can establish a new default locale to be used for all sessions on the database. You can also override individual sessions with different locales. Additionally, you can set the locale through ODBC, JDBC, and ADO.net.

Be aware of the following considerations when working with locales:

- Projections are always collated using the `en_US@collation=binary` collation, regardless of the session collation. Any locale-specific collation is applied at query run time.
- The maximum length parameter for VARCHAR and CHAR data types refers to the number of octets (bytes) that can be stored in that field, not the number of characters. When you use multi-byte UTF-8 characters, the fields must be sized to accommodate from 1 to 4 bytes per character, depending on the data.
- When the locale is non-binary, use the collation function to transform the input to a binary string that sorts in the proper order. This transformation increases the number of bytes required for the input according to this formula (CollationExpansion defaults to 5):

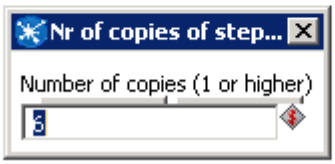
  ```
  result_column_width = input_octet_width * CollationExpansion + 4
  ```

# Enabling PDI parallelization

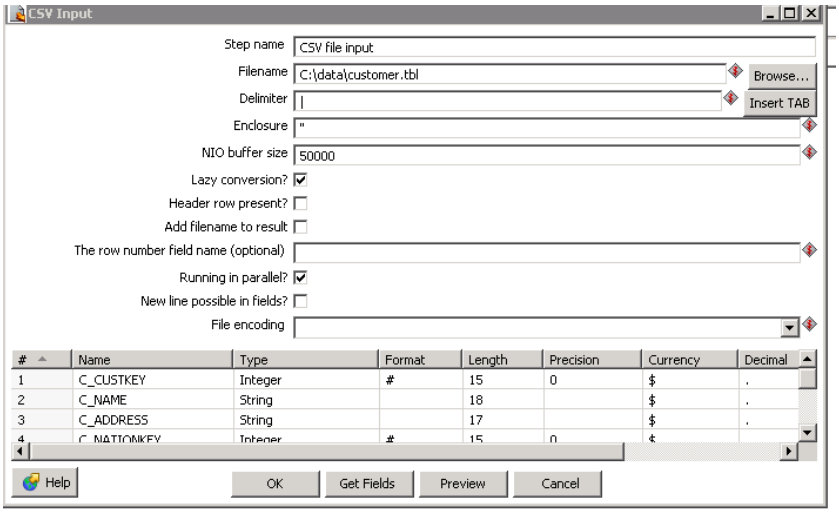
All versions of PDI include the ability to run transformations in parallel. How it works depends on the format of the data source, as described in the next two sections.

## Parallelization When the Source is a File

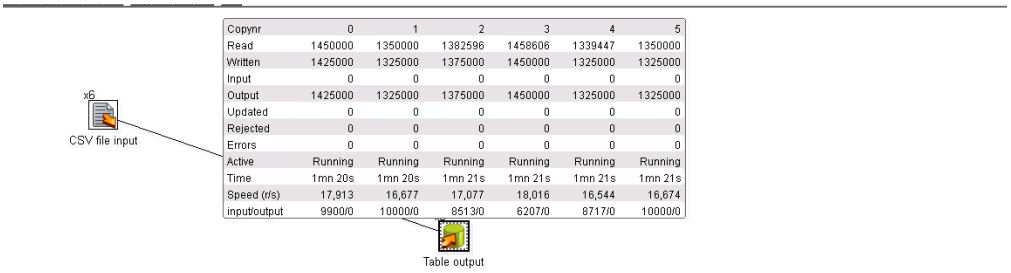
When the source is a file, to implement parallelization, right-click the input component and set the **Number of copies** value. Note the default of "distribution" is chosen for the data distribution method. Here is an example transform using the standard input CSV and the standard table output:

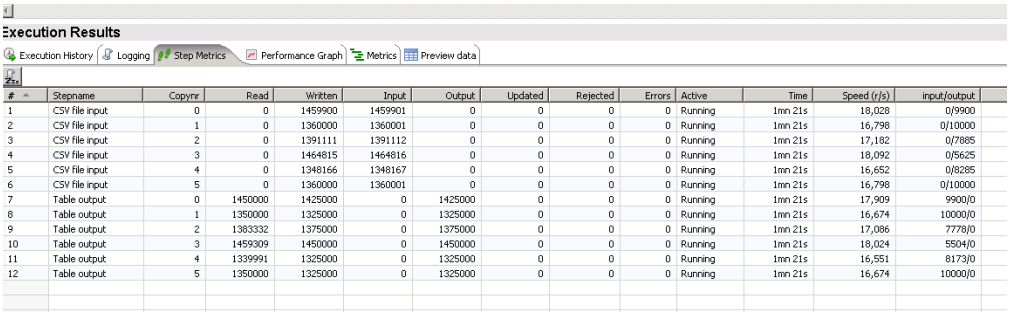




The CSV file input step *must* be configured to run in parallel. If you have not configured this capability, each copy of the step reads the full file, which creates duplicates. To configure this capability, double-click the CSV file input step and make sure that **Running in parallel?** is selected, as shown in the following figure.

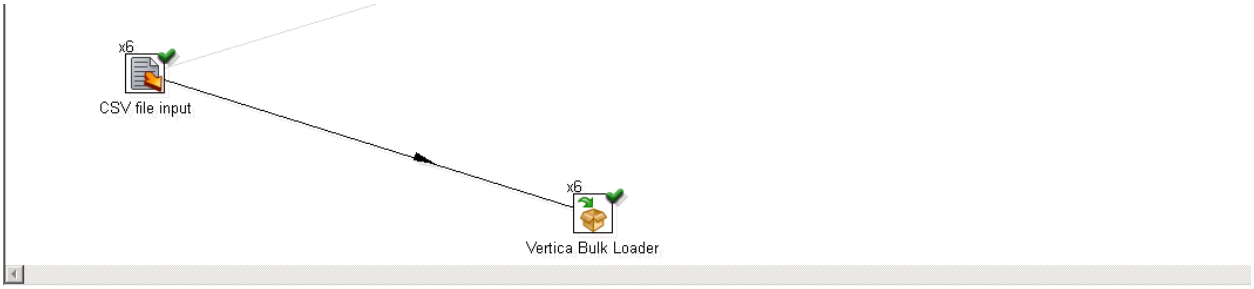



In this example, you specify 6 for the number of copies and for the output. As you can see from the output below, this parallelizes the reads and writes.




Tests indicate that when using the HP Vertica Bulk Loader component, less memory is used on the PDI machine. On average, loads are almost twice as fast, depending on the resources of your PDI machine and source table size.




**Execution Results**

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

| # | Stepname | Copynr | Read | Written | Input | Output | Updated | Rejected | Errors | Active | Time | Speed (r/s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CSV file input | 0 | 0 | 2510820 | 2510820 | 0 | 0 | 0 | 0 | Finished | 32.7s | 76,861 |
| 2 | CSV file input | 1 | 0 | 2503657 | 2503657 | 0 | 0 | 0 | 0 | Finished | 55.1s | 45,400 |
| 3 | CSV file input | 2 | 0 | 2504474 | 2504474 | 0 | 0 | 0 | 0 | Finished | 59.2s | 42,291 |
| 4 | CSV file input | 3 | 0 | 2503714 | 2503714 | 0 | 0 | 0 | 0 | Finished | 40.0s | 62,554 |
| 5 | CSV file input | 4 | 0 | 2488647 | 2488647 | 0 | 0 | 0 | 0 | Finished | 33.8s | 73,666 |
| 6 | CSV file input | 5 | 0 | 2488688 | 2488688 | 0 | 0 | 0 | 0 | Finished | 36.6s | 68,028 |
| 7 | Vertica Bulk Loader | 0 | 2510820 | 2510820 | 0 | 2510820 | 0 | 0 | 0 | Finished | 34.0s | 73,845 |
| 8 | Vertica Bulk Loader | 1 | 2503657 | 2503657 | 0 | 2503657 | 0 | 0 | 0 | Finished | 56.7s | 44,194 |
| 9 | Vertica Bulk Loader | 2 | 2504474 | 2504474 | 0 | 2504474 | 0 | 0 | 0 | Finished | 1mn 0s | 41,269 |
| 10 | Vertica Bulk Loader | 3 | 2503714 | 2503714 | 0 | 2503714 | 0 | 0 | 0 | Finished | 41.5s | 60,301 |
| 11 | Vertica Bulk Loader | 4 | 2488647 | 2488647 | 0 | 2488647 | 0 | 0 | 0 | Finished | 35.0s | 71,058 |
| 12 | Vertica Bulk Loader | 5 | 2488688 | 2488688 | 0 | 2488688 | 0 | 0 | 0 | Finished | 37.9s | 65,639 |

For more information, see this post in the Pentaho Forum.

## Parallelization When the Source is a Table

When the source is a table, implementing parallelism for the output can lead to a 30% performance boost. The following example shows how to use the Standard Table Output component. In this transformation, PDI parallelizes only the write operations.

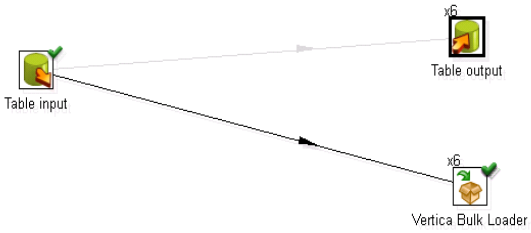

**Execution Results**

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

| # | Stepname | Copynr | Read | Written | Input | Output | Updated | Rejected | Errors | Active | Time | Speed (r/s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Table input | 0 | 0 | 15000000 | 15000000 | 0 | 0 | 0 | 0 | Finished | 6mn 42s | 37,306 |
| 2 | Table output | 0 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,182 |
| 3 | Table output | 1 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,183 |
| 4 | Table output | 2 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,182 |
| 5 | Table output | 3 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,183 |
| 6 | Table output | 4 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,183 |
| 7 | Table output | 5 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 6mn 44s | 6,183 |

The next example shows how to use the HP Vertica Bulk Loader when the source is a table. For this transformation, PDI again only parallelizes the write operations.

.




**Execution Results**

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

| # | Stepname | Copynr | Read | Written | Input | Output | Updated | Rejected | Errors | Active | Time | Speed (r/s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Table input | 0 | 0 | 15000000 | 15000000 | 0 | 0 | 0 | 0 | Finished | 5mn 32s | 45,089 |
| 2 | Vertica Bulk Loader | 0 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,485 |
| 3 | Vertica Bulk Loader | 1 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,480 |
| 4 | Vertica Bulk Loader | 2 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,482 |
| 5 | Vertica Bulk Loader | 3 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,480 |
| 6 | Vertica Bulk Loader | 4 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,483 |
| 7 | Vertica Bulk Loader | 5 | 2500000 | 2500000 | 0 | 2500000 | 0 | 0 | 0 | Finished | 5mn 34s | 7,481 |

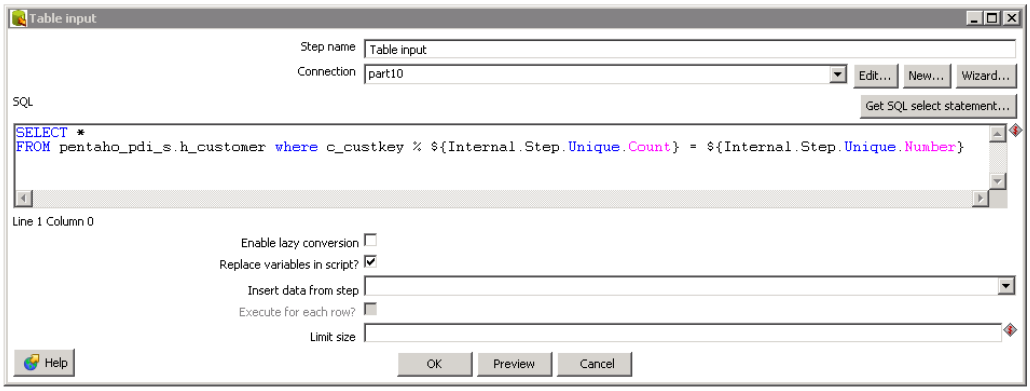
In order to achieve better performance, implement faster input **reads** by introducing SELECT logic as follows:

1. Take the original SELECT query.
2. Add a WHERE clause to chunk the data.

In the following example, the HP Vertica Bulk Loader reads the data in six chunks.

PDI provides variables (where mod(c_custkey,${Internal.Step.Unique.Count}) = ${Internal.Step.Unique.Number}) that implement this chunking easily. The following figure shows an example.

**Note:** When PDI asks you to choose a split method, select **Distribute Rows**. If you are using the PDI Enterprise Edition, you can use **Load Balancing** instead.


This example generates the following parallel queries to the source database. The WHERE clause defines the data chunking:

```
NEW:
        SELECT * FROM pentaho_pdi_s.h_customer WHERE mod(c_custkey,6) = 0;
        SELECT * FROM pentaho_pdi_s.h_customer WHERE mod(c_custkey,6) = 1;
```

```
SELECT * FROM pentaho_pdi_s.h_customer WHERE mod(c_custkey,6) = 2;
SELECT * FROM pentaho_pdi_s.h_customer WHERE mod(c_custkey,6) = 3;
SELECT * FROM pentaho_pdi_s.h_customer WHERE mod(c_custkey,6) = 4;
SELECT * FROM pentaho_pdi_s.h_customer where mod(c_custkey,6) = 5;
```

Note: `c_custkey` must be an integer and should be evenly distributed. If there are more keys that its mod is 3, the fourth copy will have more rows. The fourth copy will perform slower than the other copies.

### Reducing Parallelism to Control I/O Swap

If the swapping I/O is significant, the system is under stress. (Use `vmstat` or `iostat` to capture the I/O.) Significant I/O swapping can occur when

si + so > 20% of maximum – seen bi + bo

Thus, up to 20% of the disk bandwidth can take up over many minutes. This condition presents most often during the high block I/O periods.

In this situation, reduce the parallelism of the load, by reducing the number of load streams.

## Using Row Buffers to Troubleshoot PDI

PDI creates a row buffer between each step. Steps retrieve rows of data from their inbound row buffer, process the rows, and pass them into an outbound row buffer that feeds into the subsequent step. By default, row buffers can hold up to 10,000 rows, but this value can be configured for each transformation.

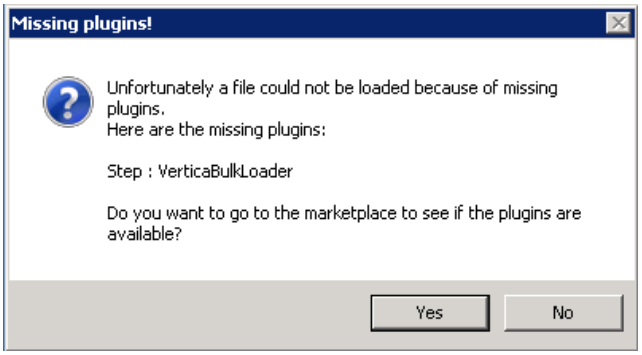
When you run a transformation, the **Step Metrics** tab on the **Execution Results** pane shows real-time statistics for each step. The input/output field shows a real-time display of the number of rows in the buffers feeding into and coming out of each step. If the input buffer of a step is full, that step cannot keep up with the rows being fed into it. For more examples, see PDI Performance Tuning Checklist.

# Known Issues

You may encounter the following issues when connecting to HP Vertica using PDI.

### Upgrade Issues

As of 5.1.0 of PDI, the HP Vertica Plugin is not in the Pentaho Marketplace for the PDI Community Edition. When doing an upgrade you may encounter the following message:




Download and unzip the HP Vertica Bulk Loader into the following directory:

`http://ci.pentaho.com/job/Kettle-VerticaBulkLoader/`

You can download the zip, and unzip it into Kettle's plugins folder:

`C:\<pdi-install-folder>\pentaho\plugins`

The HP Vertica Bulk Loader appears as a new step in the Bulk Loaders category after restarting Spoon.

### PDI HP Vertica Bulk Loader Error IllegalArgumentException

The root cause of the IllegalArgumentException error (PDI-9912) is the way that the HP Vertica Bulk Loader step allocates memory to store data for VARCHAR columns. Currently, no optimizations are done in the HP Vertica Bulk Loader to allocate just

enough memory for the data set that is pushed into HP Vertica. Instead, the HP Vertica Bulk Loader allocates the maximum of 65,000 bytes for each VARCHAR column, which is likely far beyond any length stored in the database.

The current method for calculating the memory allocation for the HP Vertica Bulk Loader is

[Max Column Size in Bytes] * 1000 rows for each column

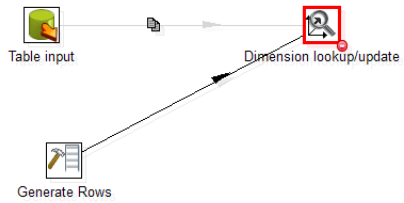
In the meantime, Hewlett Packard recommends that you increase the Java Memory allocation for Spoon, seeing if the additional memory allows the HP Vertica Bulk Loader to complete successfully. For an HP Vertica table with 20 VARCHAR columns, you should allocate 1,240 MB of memory (65,000 bytes * 20 columns * 1000 rows).

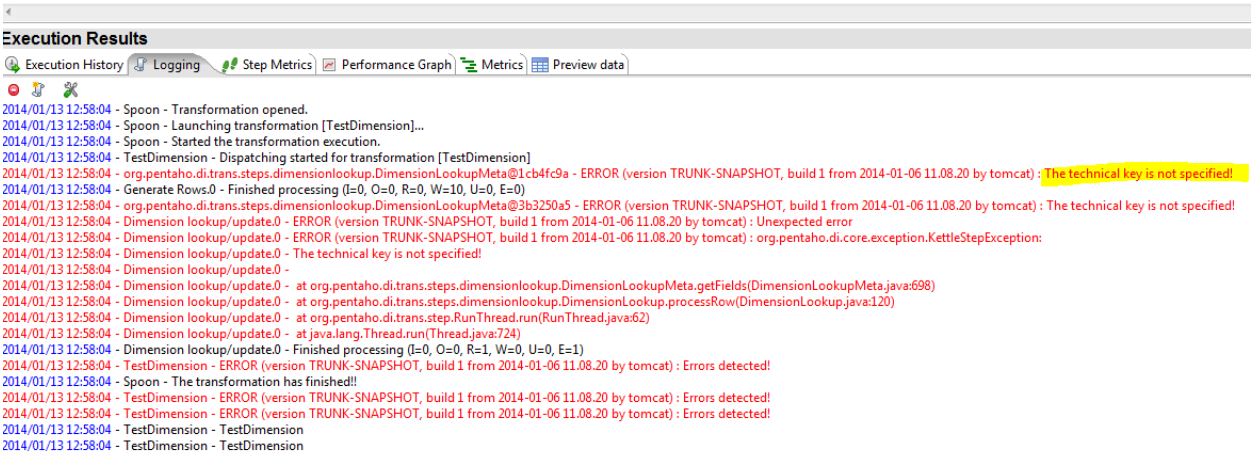
In the meantime, it is recommended to increase the Java Memory allocation

**Note:** This problem has been fixed in PDI 5.1.

## Dimension Lookup/Update Step Error

A technical key is mandatory for slowly changing dimensions and for the Dimension Lookup/Update step. In PDI-9815, the `TestDimension.ktr` technical key was not specified, and the transformation was failing. The error message has been enhanced to provide a better description of reason of step failure. The following figure shows an example of this error.




**Note:** This problem has been fixed in PDI 5.1.

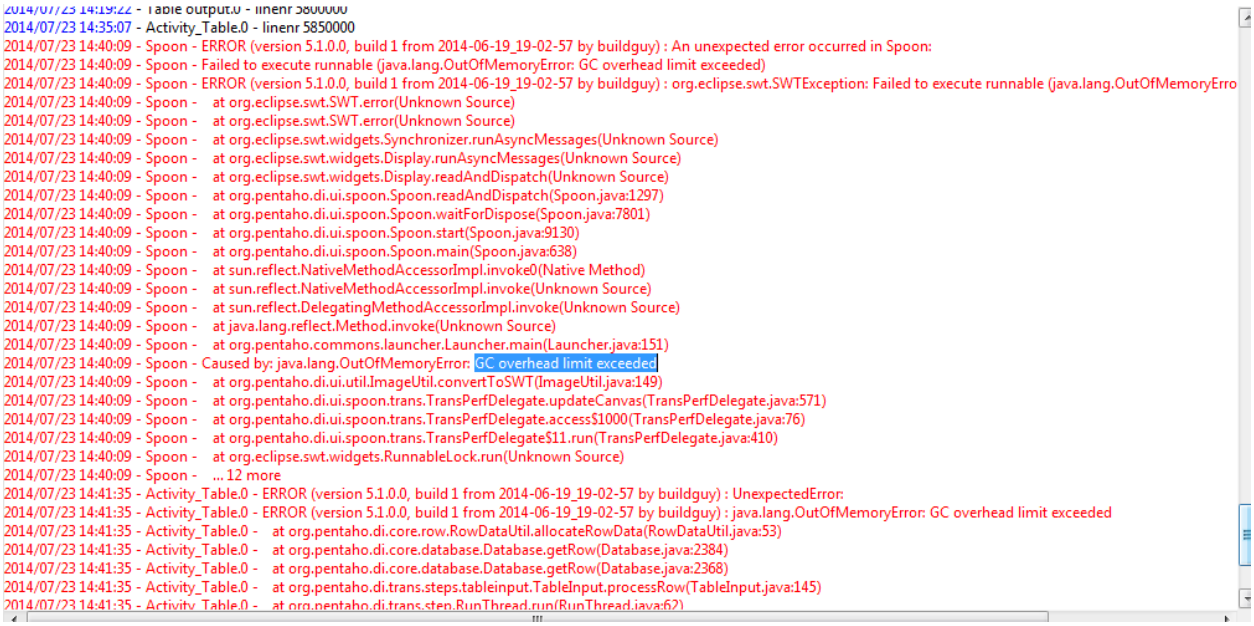
## GC Overhead Limit Exceeded Error

The default `spoon.sh` startup file is usually configured for 512 MB of RAM, as follows:

```
if "%PENTAHO_DI_JAVA_OPTIONS%"=="" set PENTAHO_DI_JAVA_OPTIONS="-Xmx512m" "-
XX:MaxPermSize=256m"
```

If you are getting an error, increase the amount of RAM allocated in the startup file. If your machine only has 4 GB of RAM, try allocating 3 GB, as in the following example:

```
if "%PENTAHO_DI_JAVA_OPTIONS%"=="" set PENTAHO_DI_JAVA_OPTIONS="-Xmx3g" "-
XX:MaxPermSize=256m"
```




### Unsupported Data Types

PDI does not support the following data types:

- TimeStampTz
- TimeTz
- Time

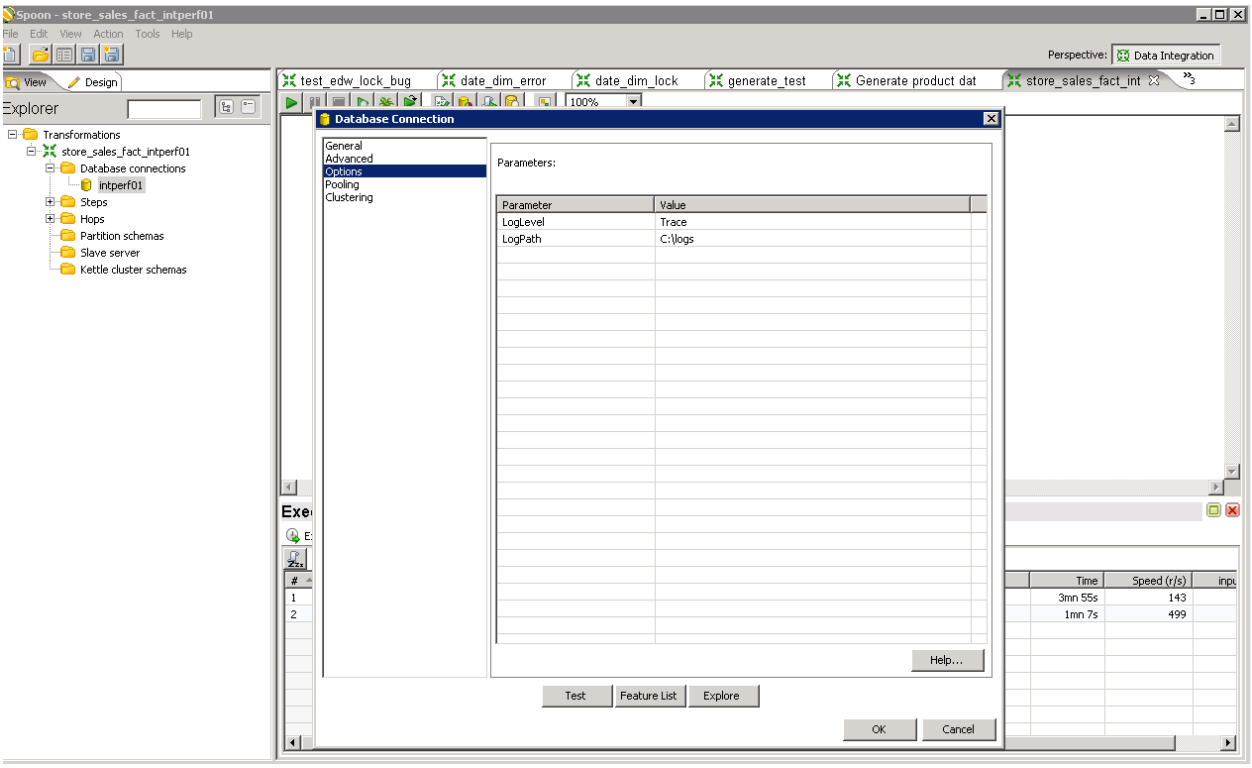
There are some alternative PDI mappings that can be done to process these data types:

- PDI converts the TimestampTz data type with Timestamp(Truncated timezone from data). When the target and source schemas are in the same time zone, this conversion works. However, it may fail if the source and target schemas are in different time zones.
- The TimeTz data type is not automatically converted. You can manually convert it to a VARCHAR.
- The Time data type is automatically converted into the Timestamp data type. You need to override the HH:mm:ss.SSS format for any Time columns.

## Tracing

To allow tracing, set the following parameters:

- LogLevel = Trace
- LogPath = Path to the log file location




## For More Information

For additional information about optimizing HP Vertica to work with PDI:

- http://www.vertica.com/customer-experience/vertica-101/
- http://www.vertica-forums.com/viewforum.php?f=11
- https://inter.viewcentral.com/events/cust/catalog.aspx?cid=arcsight&pid=1&lid=2&p_bu_id=6

Hewlett Packard provides training on this topic: http://h10120.www1.hp.com/expertone/datacard/Course/00787054