Contents

To take advantage of linked datastores..........................................................60
Example of push-down with linked datastores...........................................61
Generated SQL statements..........................................................................63
Tuning performance at the data flow or Job Server level............................63

Chapter 6 Using Caches

Caching data.................................................................................................66
Caching sources............................................................................................67
Caching joins.................................................................................................68
To change the cache type for a data flow....................................................68
Caching lookups............................................................................................69
Caching table comparisons..........................................................................70
Specifying a pageable cache directory.......................................................71
Using persistent cache................................................................................72
Using persistent cache tables as sources.....................................................72
Monitoring and tuning caches......................................................................73
Using statistics for cache self-tuning............................................................73
To automatically choose the cache type.......................................................73
To monitor and tune in-memory and pageable caches...............................73

Chapter 7 Using Parallel Execution

Parallel data flows and work flows...............................................................80
Parallel execution in data flows....................................................................81
Table partitioning........................................................................................81
Degree of parallelism...................................................................................87
Combining table partitioning and a degree of parallelism..........................95
File multi-threading....................................................................................98

Chapter 8 Distributing Data Flow Execution

Splitting a data flow into sub data flows.....................................................104
Contents

Run as a separate process option .................................................................105
Examples of multiple processes for a data flow........................................105
Data_Transfer transform ...........................................................................112
Examples of multiple processes with Data_Transfer ................................112
Using grid computing to distribute data flow execution............................118
Server Group ............................................................................................118
Distribution levels for data flow execution .................................................118

Chapter 9 Using Bulk Loading ....................................................................125

Bulk loading in Oracle .............................................................................126
Bulk-loading methods ..............................................................................126
Bulk-loading modes ................................................................................127
Bulk-loading parallel-execution options ..................................................127
Bulk-loading scenarios .............................................................................128
Using bulk-loading options .....................................................................129
Bulk loading in Microsoft SQL Server ....................................................132
To use the SQL Server ODBC bulk copy API ........................................132
Network packet size option .....................................................................132
Maximum rejects option ..........................................................................133
Bulk loading in HP Neoview ....................................................................133
How Data Services and HP Neoview use the file options to load ..........134
Using the UPSERT bulk operation ...........................................................135
Bulk loading in Informix ...........................................................................136
To set Informix server variables ..............................................................136
Bulk loading in DB2 Universal Database ................................................136
When to use each DB2 bulk-loading method .........................................137
Using the DB2 CLI load method ..............................................................138
To configure your system to use the CLI load method ...........................138
To use the CLI load method in a job .......................................................139
Using the DB2 bulk load utility ...............................................................139
To configure your system to use the load utility .....................................140
To use the load utility in a job.......................... 142
Using the import utility .................................... 144
Bulk loading in Netezza.................................... 144
Netezza bulk-loading process.......................... 145
Options overview.......................................... 145
Netezza log files............................................ 146
Configuring bulk loading for Netezza.............. 147
To configure the target table......................... 147
Bulk loading in Sybase ASE............................ 148
Bulk loading in Sybase IQ............................... 148
Bulk loading in Teradata............................... 149
When to use each Teradata bulk load method.... 150
How SAP BusinessObjects Data Services and Teradata use the file options to load........................ 157
Using the UPSERT bulk operation.................. 159
Parallel Transporter method ......................... 160
Teradata standalone utilities ......................... 164

Chapter 10   Other Tuning Techniques  179

Source-based performance options................ 181
Join ordering............................................. 181
Minimizing extracted data.......................... 184
Using array fetch size............................... 185
Target-based performance options............... 186
Loading method ....................................... 186
Rows per commit....................................... 187
Job design performance options.................. 188
Loading only changed data......................... 188
Minimizing data type conversion................ 189
Minimizing locale conversion..................... 189
Precision in operations............................. 189
Welcome to SAP BusinessObjects Data Services
Welcome

SAP BusinessObjects Data Services XI Release 3 provides data integration and data quality processes in one runtime environment, delivering enterprise performance and scalability.

The data integration processes of SAP BusinessObjects Data Services allow organizations to easily explore, extract, transform, and deliver any type of data anywhere across the enterprise.

The data quality processes of SAP BusinessObjects Data Services allow organizations to easily standardize, cleanse, and consolidate data anywhere, ensuring that end users are always working with information that's readily available, accurate, and trusted.

Documentation set for SAP BusinessObjects Data Services

You should become familiar with all the pieces of documentation that relate to your SAP BusinessObjects Data Services product.

<table>
<thead>
<tr>
<th>Document</th>
<th>What this document provides</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation Map</td>
<td>Information about available SAP BusinessObjects Data Services books, languages, and locations</td>
</tr>
<tr>
<td>Release Summary</td>
<td>Highlights of key features in this SAP BusinessObjects Data Services release</td>
</tr>
<tr>
<td>Release Notes</td>
<td>Important information you need before installing and deploying this version of SAP BusinessObjects Data Services</td>
</tr>
<tr>
<td>Getting Started Guide</td>
<td>An introduction to SAP BusinessObjects Data Services</td>
</tr>
<tr>
<td>Installation Guide for Windows</td>
<td>Information about and procedures for installing SAP BusinessObjects Data Services in a Windows environment</td>
</tr>
<tr>
<td>Document</td>
<td>What this document provides</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Installation Guide for UNIX</strong></td>
<td>Information about and procedures for installing SAP BusinessObjects Data Services in a UNIX environment</td>
</tr>
<tr>
<td><strong>Advanced Development Guide</strong></td>
<td>Guidelines and options for migrating applications including information on multi-user functionality and the use of the central repository for version control</td>
</tr>
<tr>
<td><strong>Designer Guide</strong></td>
<td>Information about how to use SAP BusinessObjects Data Services Designer</td>
</tr>
<tr>
<td><strong>Integrator's Guide</strong></td>
<td>Information for third-party developers to access SAP BusinessObjects Data Services functionality using web services and APIs</td>
</tr>
<tr>
<td><strong>Management Console: Administrator Guide</strong></td>
<td>Information about how to use SAP BusinessObjects Data Services Administrator</td>
</tr>
<tr>
<td><strong>Management Console: Metadata Reports Guide</strong></td>
<td>Information about how to use SAP BusinessObjects Data Services Metadata Reports</td>
</tr>
<tr>
<td><strong>Migration Considerations</strong></td>
<td>Release-specific product behavior changes from earlier versions of SAP BusinessObjects Data Services to the latest release. This manual also contains information about how to migrate from SAP BusinessObjects Data Quality Management to SAP BusinessObjects Data Services</td>
</tr>
<tr>
<td><strong>Performance Optimization Guide</strong></td>
<td>Information about how to improve the performance of SAP BusinessObjects Data Services</td>
</tr>
<tr>
<td><strong>Reference Guide</strong></td>
<td>Detailed reference material for SAP BusinessObjects Data Services Designer</td>
</tr>
</tbody>
</table>
A compiled “master” PDF of core SAP BusinessObjects Data Services books containing a searchable master table of contents and index:
- Getting Started Guide
- Installation Guide for Windows
- Installation Guide for UNIX
- Designer Guide
- Reference Guide
- Management Console: Metadata Reports Guide
- Management Console: Administrator Guide
- Performance Optimization Guide
- Advanced Development Guide
- Supplement for J.D. Edwards
- Supplement for Oracle Applications
- Supplement for PeopleSoft
- Supplement for Siebel
- Supplement for SAP

A step-by-step introduction to using SAP BusinessObjects Data Services

In addition, you may need to refer to several Adapter Guides and Supplemental Guides.

<table>
<thead>
<tr>
<th>Document</th>
<th>What this document provides</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Manuals</td>
<td></td>
</tr>
<tr>
<td><em>Getting Started Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Installation Guide for Windows</em></td>
<td></td>
</tr>
<tr>
<td><em>Installation Guide for UNIX</em></td>
<td></td>
</tr>
<tr>
<td><em>Designer Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Reference Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Management Console: Metadata Reports Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Management Console: Administrator Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Performance Optimization Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Advanced Development Guide</em></td>
<td></td>
</tr>
<tr>
<td><em>Supplement for J.D. Edwards</em></td>
<td></td>
</tr>
<tr>
<td><em>Supplement for Oracle Applications</em></td>
<td></td>
</tr>
<tr>
<td><em>Supplement for PeopleSoft</em></td>
<td></td>
</tr>
<tr>
<td><em>Supplement for Siebel</em></td>
<td></td>
</tr>
<tr>
<td><em>Supplement for SAP</em></td>
<td></td>
</tr>
<tr>
<td>Tutorial</td>
<td></td>
</tr>
<tr>
<td>A step-by-step introduction to using SAP BusinessObjects Data Services</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document</th>
<th>What this document provides</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce.com Adapter Interface</td>
<td>Information about how to install, configure, and use the SAP BusinessObjects Data Services Salesforce.com Adapter Interface</td>
</tr>
<tr>
<td>Supplement for J.D. Edwards</td>
<td>Information about interfaces between SAP BusinessObjects Data Services and J.D. Edwards World and J.D. Edwards OneWorld</td>
</tr>
<tr>
<td>Supplement for Oracle Applications</td>
<td>Information about the interface between SAP BusinessObjects Data Services and Oracle Applications</td>
</tr>
<tr>
<td>Supplement for PeopleSoft</td>
<td>Information about interfaces between SAP BusinessObjects Data Services and PeopleSoft</td>
</tr>
<tr>
<td>Document</td>
<td>What this document provides</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Supplement for SAP</strong></td>
<td>Information about interfaces between SAP BusinessObjects Data Services, SAP Applications, and SAP NetWeaver BW</td>
</tr>
<tr>
<td><strong>Supplement for Siebel</strong></td>
<td>Information about the interface between SAP BusinessObjects Data Services and Siebel</td>
</tr>
</tbody>
</table>

## Accessing documentation

You can access the complete documentation set for SAP BusinessObjects Data Services in several places.

### Accessing documentation on Windows

After you install SAP BusinessObjects Data Services, you can access the documentation from the Start menu.

1. Choose **Start > Programs > SAP BusinessObjects XI 3.2 > SAP BusinessObjects Data Services > Data Services Documentation**.
   
   **Note:**
   Only a subset of the documentation is available from the Start menu. The documentation set for this release is available in `LINK_DIR\Doc\Books\en`.

2. Click the appropriate shortcut for the document that you want to view.

### Accessing documentation on UNIX

After you install SAP BusinessObjects Data Services, you can access the online documentation by going to the directory where the printable PDF files were installed.

2. Using Adobe Reader, open the PDF file of the document that you want to view.
Accessing documentation from the Web

You can access the complete documentation set for SAP BusinessObjects Data Services from the SAP BusinessObjects Technical Customer Assurance site.

2. Click SAP BusinessObjects at the top of the page.
3. Click All Products in the navigation pane on the left.

You can view the PDFs online or save them to your computer.

SAP BusinessObjects information resources

A global network of SAP BusinessObjects technology experts provides customer support, education, and consulting to ensure maximum business intelligence benefit to your business.

Useful addresses at a glance:

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
</table>

Welcome to SAP BusinessObjects Data Services
SAP BusinessObjects information resources
<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
</table>
| Customer Support, Consulting, and Education services  
http://service.sap.com/ | Information about Technical Customer Assurance programs, as well as links to technical articles, downloads, and online forums. Consulting services can provide you with information about how SAP BusinessObjects can help maximize your business intelligence investment. Education services can provide information about training options and modules. From traditional classroom learning to targeted e-learning seminars, SAP BusinessObjects can offer a training package to suit your learning needs and preferred learning style. |
| SAP BusinessObjects Data Services Community  
https://www.sdn.sap.com/irj/boc/ds | Get online and timely information about SAP BusinessObjects Data Services, including tips and tricks, additional downloads, samples, and much more. All content is to and from the community, so feel free to join in and contact us if you have a submission. |
| Forums on SCN (SAP Community Network)  
https://www.sdn.sap.com/irj/scn/forums | Search the SAP BusinessObjects forums on the SAP Community Network to learn from other SAP BusinessObjects Data Services users and start posting questions or share your knowledge with the community. |
| Blueprints  
http://www.sdn.sap.com/irj/boc/blueprints | Blueprints for you to download and modify to fit your needs. Each blueprint contains the necessary SAP BusinessObjects Data Services project, jobs, data flows, file formats, sample data, template tables, and custom functions to run the data flows in your environment with only a few modifications. |
| Product documentation  
| Products Availability Report  
https://service.sap.com/bosap-support | Get information about supported platforms for SAP BusinessObjects Data Services.  
In the left panel of the window, navigate to Documentation > Supported Platforms > BusinessObjects XI 3.x. Click the BusinessObjects Data Services link in the main window. |
Environment Test Strategy
This section covers suggested methods of tuning source and target database applications, their operating systems, and the network used by your SAP BusinessObjects Data Services environment. It also introduces key job execution options.

This section contains the following topics:

- *The source OS and database server* on page 18
- *The target OS and database server* on page 19
- *The network* on page 20
- *Job Server OS and job options* on page 20

To test and tune jobs, work with all four of these components in the order shown above.

In addition to the information in this section, you can use your UNIX or Windows operating system and database server documentation for specific techniques, commands, and utilities that can help you measure and tune the SAP BusinessObjects Data Services environment.

## The source OS and database server

Tune the source operating system and database to quickly read data from disks.

### Operating system

Make the input and output (I/O) operations as fast as possible. The read-ahead protocol, offered by most operating systems, can greatly improve performance. This protocol allows you to set the size of each I/O operation. Usually its default value is 4 to 8 kilobytes which is too small. Set it to at least 64K on most platforms.

### Database

Tune your database on the source side to perform SELECTs as quickly as possible.
In the database layer, you can improve the performance of SELECTs in several ways, such as the following:

- Create indexes on appropriate columns, based on your data flows.
- Increase the size of each I/O from the database server to match the OS read-ahead I/O size.
- Increase the size of the shared buffer to allow more data to be cached in the database server.
- Cache tables that are small enough to fit in the shared buffer. For example, if jobs access the same piece of data on a database server, then cache that data. Caching data on database servers will reduce the number of I/O operations and speed up access to database tables.

See your database server documentation for more information about techniques, commands, and utilities that can help you measure and tune the source databases in your jobs.

The target OS and database server

Tune the target operating system and database to quickly write data to disks.

Operating system

Make the input and output operations as fast as possible. For example, the asynchronous I/O, offered by most operating systems, can greatly improve performance. Turn on the asynchronous I/O.

Database

Tune your database on the target side to perform INSERTs and UPDATES as quickly as possible.

In the database layer, there are several ways to improve the performance of these operations.

Here are some examples from Oracle:

- Turn off archive logging
• Turn off redo logging for all tables
• Tune rollback segments for better performance
• Place redo log files and data files on a raw device if possible
• Increase the size of the shared buffer

See your database server documentation for more information about techniques, commands, and utilities that can help you measure and tune the target databases in your jobs.

The network

When reading and writing data involves going through your network, its ability to efficiently move large amounts of data with minimal overhead is very important. Do not underestimate the importance of network tuning (even if you have a very fast network with lots of bandwidth).

Set network buffers to reduce the number of round trips to the database servers across the network. For example, adjust the size of the network buffer in the database client so that each client request completely fills a small number of network packets.

Job Server OS and job options

Tune the Job Server operating system and set job execution options to improve performance and take advantage of self-tuning features of SAP BusinessObjects Data Services.

Operating system

SAP BusinessObjects Data Services jobs are multi-threaded applications. Typically a single data flow in a job initiates one al_engine process that in turn initiates at least 4 threads.

For maximum performance benefits:
• Consider a design that will run one al_engine process per CPU at a time.
• Tune the Job Server OS so that threads spread to all available CPUs.

For more information, see *Checking system utilization* on page 26.

## Jobs

You can tune job execution options after:

- Tuning the database and operating system on the source and the target computers
- Adjusting the size of the network buffer
- Your data flow design seems optimal

You can tune the following execution options to improve the performance of your jobs:

- **Monitor sample rate**
- **Collect statistics for optimization** and **Use collected statistics**

### Setting Monitor sample rate

During job execution, the SAP BusinessObjects Data Services writes information to the monitor log file and updates job events after processing the number of rows specified in **Monitor sample rate**. Default value is 1000. Increase **Monitor sample rate** to reduce the number of calls to the operating system to write to the log file.

When setting **Monitor sample rate**, you must evaluate performance improvements gained by making fewer calls to the operating system against your ability to view more detailed statistics during job execution. With a higher **Monitor sample rate**, the software collects more data before calling the operating system to open the file, and performance improves. However, with a higher monitor rate, more time passes before you can view statistics during job execution.

In production environments when your jobs transfer large volumes of data, you should increase **Monitor sample rate** to 50,000.
Note:
If you use a virus scanner on your files, exclude the SAP BusinessObjects Data Services log from the virus scan. Otherwise, the virus scan analyzes the log repeatedly during the job execution, which causes a performance degradation.

Collecting statistics for self-tuning

SAP BusinessObjects Data Services provides a self-tuning feature to determine the optimal cache type (in-memory or pageable) to use for a data flow.

To take advantage of this self-tuning feature

1. When you first execute a job, select the option Collect statistics for optimization to collect statistics which include number of rows and width of each row. Ensure that you collect statistics with data volumes that represent your production environment. This option is not selected by default.
2. The next time you execute the job, this option is selected by default.
3. When changes occur in data volumes, re-run your job with Collect statistics for optimization to ensure that the software has the most current statistics to optimize cache types.

For more information about these caches, see .

Related Topics
• Using Caches on page 65
Measuring Performance
This section contains the following topics:

- *Data Services processes and threads* on page 24
- *Measuring performance of jobs* on page 25

## Data Services processes and threads

Data Services uses processes and threads to execute jobs that extract data from sources, transform the data, and load data into a data warehouse. The number of concurrently executing processes and threads affects the performance of Data Services jobs.

### Processes

The processes used to run jobs are:

- **al_jobserver**
  
The al_jobserver initiates one process for each Job Server configured on a computer. This process does not use much CPU power because it is only responsible for launching each job and monitoring the job's execution.

- **al_engine**
  
  For batch jobs, an al_engine process runs when a job starts and for each of its data flows. Real-time jobs run as a single process.

  The number of processes a batch job initiates also depends upon the number of:

  - parallel work flows
  - parallel data flows
  - sub data flows

  For an example of the monitor log that displays the processes, see *Analyzing log files for task duration* on page 30.
Threads

A data flow typically initiates one al_engine process, which creates one thread per data flow object. A data flow object can be a source, transform, or target. For example, two sources, a query, and a target could initiate four threads.

If you are using parallel objects in data flows, the thread count will increase to approximately one thread for each source or target table partition. If you set the Degree of parallelism (DOP) option for your data flow to a value greater than one, the thread count per transform will increase. For example, a DOP of 5 allows five concurrent threads for a Query transform. To run objects within data flows in parallel, use the following features:

- Table partitioning
- File multithreading
- Degree of parallelism for data flows

Related Topics

- Using Parallel Execution on page 79

Measuring performance of jobs

You can use several techniques to measure performance of SAP BusinessObjects Data Services jobs:

- Checking system utilization on page 26
- Analyzing log files for task duration on page 30
- Reading the Monitor Log for execution statistics on page 31
- Reading the Performance Monitor for execution statistics on page 32
- Reading Operational Dashboards for execution statistics on page 34
Checking system utilization

The number of processes and threads concurrently executing affects the utilization of system resources (see Data Services processes and threads on page 24).

Check the utilization of the following system resources:

• CPU
• Memory
• Disk
• Network

To monitor these system resources, use the following tools:

For UNIX:
• top or a third party utility (such as glance for HPUX)

For Windows:
• Performance tab on the Task Manager

Depending on the performance of your jobs and the utilization of system resources, you might want to adjust the number of processes and threads. The following sections describe different situations and suggests features to adjust the number of processes and threads for each situation.

CPU utilization

SAP BusinessObjects Data Services is designed to maximize the use of CPUs and memory available to run the job.

The total number of concurrent threads a job can run depends upon job design and environment. Test your job while watching multi-threaded processes to see how much CPU and memory the job requires. Make needed adjustments to your job design and environment and test again to confirm improvements.
For example, if you run a job and see that the CPU utilization is very high, you might decrease the DOP value or run less parallel jobs or data flows. Otherwise, CPU thrashing might occur.

For another example, if you run a job and see that only half a CPU is being used, or if you run eight jobs on an eight-way computer and CPU usage is only 50%, you can be interpret this CPU utilization in several ways:

- One interpretation might be that the software is able to push most of the processing down to source and/or target databases.

- Another interpretation might be that there are bottlenecks in the database server or the network connection. Bottlenecks on database servers do not allow readers or loaders in jobs to use Job Server CPUs efficiently.

To determine bottlenecks, examine:

- Disk service time on database server computers

  Disk service time typically should be below 15 milliseconds. Consult your server documentation for methods of improving performance. For example, having a fast disk controller, moving database server log files to a raw device, and increasing log size could improve disk service time.

- Number of threads per process allowed on each database server operating system. For example:
  
  - On HPUX, the number of kernel threads per process is configurable. The CPU to thread ratio defaults to one-to-one. It is recommended that you set the number of kernel threads per CPU to between 512 and 1024.
  
  - On Solaris and AIX, the number of threads per process is not configurable. The number of threads per process depends on system resources. If a process terminates with a message like "Cannot create threads," you should consider tuning the job.

  For example, use the Run as a separate process option to split a data flow or use the Data_Transfer transform to create two sub data flows to execute sequentially. Since each sub data flow is executed by a different al_engine process, the number of threads needed for each will be 50% less than in your previous job design.
If you are using the **Degree of parallelism** option in your data flow, reduce the number for this option in the data flow Properties window.

- **Network connection speed**
  
  Determine the rate that your data is being transferred across your network.
  
  - If the network is a bottle neck, you might change your job execution distribution level from sub data flow to data flow or job to execute the entire data flow on the local Job Server.
  
  - If the capacity of your network is much larger, you might retrieve multiple rows from source databases using fewer requests.

  - Yet another interpretation might be that the system is under-utilized. In this case, you might increase the value for the **Degree of parallelism** option and increase the number of parallel jobs and data flows.

**Related Topics**

- *Using Parallel Execution* on page 79
- *Using grid computing to distribute data flow execution* on page 118
- *Using array fetch size* on page 185

**Memory**

For memory utilization, you might have one of the following different cases:

- Low amount of physical memory.

  In this case, you might take one of the following actions:
  
  - Add more memory to the Job Server.
  
  - Redesign your data flow to run memory-consuming operations in separate sub data flows that each use a smaller amount of memory, and distribute the sub data flows over different Job Servers to access memory on multiple machines. For more information, see *Splitting a data flow into sub data flows* on page 104.

  - Redesign your data flow to push down memory-consuming operations to the database. For more information, see *Push-down operations* on page 42.
For example, if your data flow reads data from a table, joins it to a file, and then groups the data to calculate an average, the group by operation might be occurring in memory. If you stage the data after the join and before the group by into a database on a different computer, then when a sub data flow reads the staged data and continues with the group processing, it can utilize memory from the database server on a different computer. This situation optimizes your system as a whole.

For information about how to stage your data, see *Data Transfer transform* on page 112. For more information about distributing sub data flows to different computers, see *Using grid computing to distribute data flow execution* on page 118.

• Large amount of memory but it is under-utilized.

In this case, you might cache more data. Caching data can improve the performance of data transformations because it reduces the number of times the system must access the database.

There are two types of caches available: in-memory and pageable. For more information, see *Caching data* on page 66.

• Paging occurs.

Pageable cache is the default cache type for data flows. On Windows and Linux, the virtual memory available to the al_engine process is 1.5 gigabytes (500 megabytes of virtual memory is reserved for other engine operations, totaling 2GB). On UNIX, the virtual memory is limited for the al_engine process to 3.5 gigabytes (500MB is reserved for other engine operations, totaling 4GB). If more memory is needed than these virtual memory limits, the software starts paging to continue executing the data flow.

If your job or data flow requires more memory than these limits, you can take advantage of one of the following features to avoid paging:

• Split the data flow into sub data flows that can each use the amount of memory set by the virtual memory limits.

Each data flow or each memory-intensive operation within a data flow can run as a separate process that uses separate memory from each other to improve performance and throughput. For more information, see *Splitting a data flow into sub data flows* on page 104.
• Push-down memory-intensive operations to the database server so that less memory is used on the Job Server computer. For more information, see Push-down operations on page 42.

Analyzing log files for task duration

The trace log shows the progress of an execution through each component (object) of a job. The following sample Trace log shows a separate Process ID (Pid) for the Job, data flow, and each of the two sub data flows.

<table>
<thead>
<tr>
<th>Pid</th>
<th>Tid</th>
<th>Type</th>
<th>Time Stamp</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>5696</td>
<td>5964</td>
<td>JCB</td>
<td>2/11/2007 11:56:37 PM</td>
<td>Reading job &lt;3F7791_1426_4b56_B8C1_919a0633f8de&gt; from the repository: S11.7.0.0000.</td>
</tr>
<tr>
<td>5696</td>
<td>5964</td>
<td>JCB</td>
<td>2/11/2007 11:56:37 PM</td>
<td>Current directory of job &lt;3F7791_1426_4b56_B8C1_919a0633f8de&gt; is &lt;C:\Pro&gt;</td>
</tr>
<tr>
<td>5696</td>
<td>5964</td>
<td>JCB</td>
<td>2/11/2007 11:56:37 PM</td>
<td>Starting job on job server host &lt;S3-VANLASQ1&gt;, port &lt;3800&gt;</td>
</tr>
<tr>
<td>5696</td>
<td>5964</td>
<td>JCB</td>
<td>2/11/2007 11:56:37 PM</td>
<td>Job &lt;Group_Orders_Job&gt; of runId &lt;200702123563756955964&gt; is initiated by &lt;5437&gt;</td>
</tr>
<tr>
<td>1252</td>
<td>4044</td>
<td>DFCOMM</td>
<td>2/11/2007 11:56:39 PM</td>
<td>Starting sub data flow &lt;Group_Orders_DF_1&gt; on job server host &lt;S3-VANLASQ1&gt;</td>
</tr>
<tr>
<td>1604</td>
<td>984</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:42 PM</td>
<td>Process to execute sub data flow &lt;Group_Orders_DF_1&gt; is started.</td>
</tr>
<tr>
<td>1604</td>
<td>9764</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:43 PM</td>
<td>Cache statistics determined that sub data flow &lt;Group_Orders_DF_1&gt; uses &lt;2&gt;</td>
</tr>
<tr>
<td>1604</td>
<td>9764</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:43 PM</td>
<td>less than (or equal to) the virtual memory &lt;154277280&gt; bytes available for each</td>
</tr>
<tr>
<td>1604</td>
<td>9764</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:43 PM</td>
<td>CACHE.</td>
</tr>
<tr>
<td>1604</td>
<td>984</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:45 PM</td>
<td>Process to execute sub data flow &lt;Group_Orders_DF_1&gt; is completed.</td>
</tr>
<tr>
<td>1252</td>
<td>4044</td>
<td>DFCOMM</td>
<td>2/11/2007 11:56:45 PM</td>
<td>Starting sub data flow &lt;Group_Orders_DF_2&gt; on job server host &lt;S3-VANLASQ1&gt;</td>
</tr>
<tr>
<td>5648</td>
<td>4784</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:49 PM</td>
<td>Cache statistics determined that sub data flow &lt;Group_Orders_DF_2&gt; uses &lt;0&gt;</td>
</tr>
<tr>
<td>5648</td>
<td>4784</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:49 PM</td>
<td>than (or equal to) the virtual memory &lt;1610612736&gt; bytes available for caches:</td>
</tr>
<tr>
<td>5648</td>
<td>4784</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:49 PM</td>
<td>MEMORY.</td>
</tr>
<tr>
<td>5648</td>
<td>5064</td>
<td>DATAPLWF</td>
<td>2/11/2007 11:56:49 PM</td>
<td>Process to execute sub data flow &lt;Group_Orders_DF_2&gt; is completed.</td>
</tr>
<tr>
<td>1252</td>
<td>4044</td>
<td>DFCOMM</td>
<td>2/11/2007 11:56:49 PM</td>
<td>Process to execute sub data flow &lt;Group_Orders_DF&gt; is completed.</td>
</tr>
</tbody>
</table>
This sample log contains messages about sub data flows, caches, and statistics.

**Related Topics**
- *Splitting a data flow into sub data flows* on page 104
- *Caching data* on page 66
- *Reference Guide: Objects, Log*

**Reading the Monitor Log for execution statistics**

The Monitor log file indicates how many rows SAP BusinessObjects Data Services produces or loads for a job. By viewing this log during job execution, you can observe the progress of row-counts to determine the location of bottlenecks. You can use the Monitor log to answer questions such as the following:

- What transform is running at any moment?
- How many rows have been processed so far?

  The frequency that the Monitor log refreshes the statistics is based on Monitor sample rate.

- How long does it take to build the cache for a lookup or comparison table? How long does it take to process the cache?

  If take long time to build the cache, use persistent cache.

- How long does it take to sort?

  If take long time to sort, you can redesign your data flow to push down the sort operation to the database.

- How much time elapses before a blocking operation sends out the first row?

  If your data flow contains resource-intensive operations after the blocking operation, you can add Data_Transfer transforms to push-down the resource-intensive operations.

You can view the Monitor log from the following tools:

- The Designer, as the job executes, when you click the Monitor icon.
The Administrator of the Management Console, when you click the Monitor link for a job from the Batch Job Status page.

The following sample Monitor log in the Designer shows the path for each object in the job, the number of rows processed and the elapsed time for each object. The Absolute time is the total time from the start of the job to when the software completes the execution of the data flow object.

<table>
<thead>
<tr>
<th>Path Name</th>
<th>State</th>
<th>Row Count</th>
<th>Elapsed Time (sec)</th>
<th>Absolute Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Group_Orders_DF_1/Pivot</td>
<td>STOP</td>
<td>830</td>
<td>0.040</td>
<td>9.721</td>
</tr>
<tr>
<td>/Group_Orders_DF_1/Data_Transfer_Query_Lookup_TRANS2</td>
<td>STOP</td>
<td>830</td>
<td>0.381</td>
<td>9.731</td>
</tr>
<tr>
<td>/Group_Orders_DF_1/ORDERS</td>
<td>STOP</td>
<td>830</td>
<td>0.050</td>
<td>9.721</td>
</tr>
<tr>
<td>/Group_Orders_DF_1/Query_Lookup</td>
<td>STOP</td>
<td>830</td>
<td>0.040</td>
<td>9.721</td>
</tr>
<tr>
<td>/Group_Orders_DF_2/Query.ORDER.AGG</td>
<td>STOP</td>
<td>35</td>
<td>0.000</td>
<td>14.346</td>
</tr>
<tr>
<td>/Group_Orders_DF_2/Query</td>
<td>STOP</td>
<td>35</td>
<td>0.000</td>
<td>14.326</td>
</tr>
</tbody>
</table>

Related Topics

- Setting Monitor sample rate on page 21
- Using persistent cache on page 72
- Push-down operations on page 42
- Data_Transfer transform for push-down operations on page 51
- Reference Guide: Objects, Log

Reading the Performance Monitor for execution statistics

The Performance Monitor displays execution information for each work flow, data flow, and sub data flow within a job. You can display the execution times in a graph or a table format. You can use the Performance Monitor to answer questions such as the following:

- Which data flows might be bottlenecks?
- How much time did a data flow or sub data flow take to execute?
- How many rows did the data flow or sub data flow process?
How much memory did a specific data flow use?

**Note:**
Memory statistics (Cache Size column) display in the Performance Monitor only if you select the *Collect statistics for monitoring* option when you execute the job.

The following sample Performance Monitor shows the following information:

- The Query_Lookup transform used 110 kilobytes of memory.
- The first sub data flow processed 830 rows, and the second sub data flow processed 35 rows.

### To view the Performance Monitor

1. Access the Management Console with one of the following methods:
   - In the Designer top menu bar, click **Tools** and select **Management Console**.
   - Click **Start > Programs > SAP BusinessObjects XI 3.2 > SAP BusinessObjects Data Services > Management Console**.
2. On the launch page, click **Administrator**.
3. Select **Batch > repository**
4. On the Batch Job Status page, find a job execution instance.
5. Under **Job Information** for an instance, click **Performance > Monitor**.

**Related Topics**
- *To monitor and tune in-memory and pageable caches* on page 73

### Reading Operational Dashboards for execution statistics

Operational dashboard reports contain job and data flow execution information for one or more repositories over a given time period (for example the last day or week). You can use operational statistics reports to answer some of the following questions:

- Are jobs executing within the allotted time frames?
- How many jobs succeeded or failed over a given execution period?
- How is the execution time for a job evolving over time?
- How many rows did the data flow process?

### To compare execution times for the same job over time

1. Open the Management Console via one of the following methods:
   - In the Designer top menu bar, click **Tools** and select **Management Console**.
   - Choose **Start > Programs > SAP BusinessObjects XI 3.2 > SAP BusinessObjects Data Services > Data Services Management Console** menu, click .
2. On the launch page, click **Operational Dashboard**.
3. Look at the graphs in Job Execution Statistic History or Job Execution Duration History to see if performance is increasing or decreasing.
4. On the Job Execution Duration History page, if there is a specific day that looks high or low compared to the other execution times, click that point on the graph to view the Job Execution Duration graph for all of the jobs that ran that day.
5. Click **View all history** to compare different executions of a specific job or data flow.

6. On the Job Execution History tab, you can select a specific job and number of days.

7. On the Data Flow Execution History tab, you can select a specific job and number of days, as well as search for a specific data flow.
Measuring performance of jobs

Related Topics

- Management Console Metadata Reports Guide: Operational Dashboard Reports
Tuning Overview
This section presents an overview of the different Data Services tuning options, with cross-references to subsequent chapters for more details.

Strategies to execute jobs

To maximize performance of your jobs, use the following tuning strategies:

- *Maximizing push-down operations to the database server* on page 38
- *Improving throughput* on page 39
- *Using advanced tuning options* on page 40

Maximizing push-down operations to the database server

SAP BusinessObjects Data Services generates SQL SELECT statements to retrieve the data from source databases. The software automatically distributes the processing workload by pushing down as much as possible to the source database server.

Pushing down operations provides the following advantages:

- Use the power of the database server to execute SELECT operations (such as joins, Group By, and common functions such as decode and string functions). Often the database is optimized for these operations.
- Minimize the amount of data sent over the network. Less rows can be retrieved when the SQL statements include filters or aggregations.

You can also do a full push down from the source to the target, which means the software sends SQL INSERT INTO... SELECT statements to the target database. The following features enable a full push down:

- Data_Transfer transform
- Database links and linked datastores

Related Topics

- *Maximizing Push-Down Operations* on page 41
Improving throughput

Use the following features to improve throughput:

- Using caches for faster access to data

  You can improve the performance of data transformations by caching as much data as possible. By caching data in memory, you limit the number of times the system must access the database.

- Bulk loading to the target

  The software supports database bulk loading engines including the Oracle bulk load API. You can have multiple bulk load processes running in parallel.

- Other tuning techniques

  - Source-based performance options
    - Join ordering
    - Minimizing extracted data
    - Using array fetch size
  
  - Target-based performance options
    - Loading method
    - Rows per commit
  
  - Job design performance options
    - Loading only changed data
    - Minimizing data type conversion
    - Minimizing locale conversion
    - Precision in operations

Related Topics

- Using Caches on page 65
- Using Bulk Loading on page 125
- Other Tuning Techniques on page 179
Using advanced tuning options

If your jobs have CPU-intensive and memory-intensive operations, you can use the following advanced tuning features to improve performance:

• Parallel processes - Individual work flows and data flows can execute in parallel if you do not connect them in the Designer workspace.

• Parallel threads - The software supports partitioned source tables, partitioned target tables, and degree of parallelism. These options allow you to control the number of instances for a source, target, and transform that can run in parallel within a data flow. Each instance runs as a separate thread and can run on a separate CPU.

• Server groups and distribution levels - You can group Job Servers on different computers into a logical component called a server group. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the computer with the lightest load at runtime. This functionality also provides a hot backup method. If one Job Server in a server group is down, another Job Server in the group processes the job.

You can distribute the execution of data flows or sub data flows within a batch job across multiple Job Servers within a Server Group to better balance resource-intensive operations.

Related Topics
• Using Parallel Execution on page 79
• Management Console Administrator Guide: Server Groups
• Using grid computing to distribute data flow execution on page 118
Maximizing Push-Down Operations
For SQL sources and targets, SAP BusinessObjects Data Services creates database-specific SQL statements based on the data flow diagrams in a job. The software generates SQL SELECT statements to retrieve the data from source databases. To optimize performance, the software pushes down as many SELECT operations as possible to the source database and combines as many operations as possible into one request to the database. It can push down SELECT operations such as joins, Group By, and common functions such as decode and string functions.

Data flow design influences the number of operations that the software can push to the database. Before running a job, you can view the SQL that is generated and adjust your design to maximize the SQL that is pushed down to improve performance.

You can use database links and the Data_Transfer transform to pushdown more operations.

This section discusses:
- Push-down operations
- Push-down examples
- Viewing SQL
- Data_Transfer transform for push-down operations
- Database link support for push-down operations across datastores

### Push-down operations

By pushing down operations to the source database, Data Services reduces the number of rows and operations that the engine must retrieve and process, which improves performance. When determining which operations to push to the database, Data Services examines the database and its environment.

### Full push-down operations

The Optimizer always first tries to do a full push-down operation. A full push-down operation is when all transform operations can be pushed down to the databases and the data streams directly from the source database to the target database. SAP BusinessObjects Data Services sends SQL INSERT

```sql
INSERT INTO target_table
SELECT * FROM source_table
WHERE condition
```
INTO... SELECT statements to the target database where SELECT retrieves data from the source.

The software does a full push-down operation to the source and target databases when the following conditions are met:

• All of the operations between the source table and target table can be pushed down.

• The source and target tables are from the same datastore or they are in datastores that have a database link defined between them.

To enable a full push-down from the source to the target, you can also use the following features:

• Data_Transfer transform

• Database links

For database targets that support the Allow Merge option, when all other operations in the data flow can be pushed down to the source database, the auto-correct loading operation may also be pushed down for a full push-down operation to the target. The software sends an SQL MERGE INTO target statement that implements the Ignore columns with value and Ignore columns with null options.

Partial push-down operations

When a full push-down operation is not possible, SAP BusinessObjects Data Services still pushes down the SELECT statement to the source database. Operations within the SELECT statement that the software can push to the database include:

• Aggregations — Aggregate functions, typically used with a Group by statement, always produce a data set smaller than or the same size as the original data set.

• Distinct rows — When you select Distinct rows from the Select tab in the query editor, the software will only output unique rows.

• Filtering — Filtering can produce a data set smaller than or equal to the original data set.
• Joins — Joins typically produce a data set smaller than or similar in size to the original tables. The software can push down joins when either of the following conditions exist:
  • The source tables are in the same datastore
  • The source tables are in datastores that have a database link defined between them

• Ordering — Ordering does not affect data-set size. The software can efficiently sort data sets that fit in memory. It is recommended that you push down the Order By for very large data sets.

• Projection — Projection is the subset of columns that you map on the Mapping tab in the query editor. Projection normally produces a smaller data set because it only returns columns needed by subsequent operations in a data flow.

• Functions — Most functions that have equivalents in the underlying database are appropriately translated. These functions include decode, aggregation, and string functions.

### Operations that cannot be pushed down

SAP BusinessObjects Data Services cannot push some transform operations to the database. For example:

• Expressions that include functions that do not have database correspondents

• Load operations that contain triggers

• Transforms other than Query

• Joins between sources that are on different database servers that do not have database links defined between them.

Similarly, the software cannot always combine operations into single requests. For example, when a stored procedure contains a COMMIT statement or does not return a value, the software cannot combine the stored procedure SQL with the SQL for other operations in a query.

The software can only push operations supported by the DBMS down to that DBMS. Therefore, for best performance, try not to intersperse SAP
BusinessObjects Data Services transforms among operations that can be pushed down to the database.

Push-down examples

The following are typical push-down scenarios.

Collapsing transforms to push down operations example

When determining how to push operations to the database, SAP BusinessObjects Data Services first collapses all the transforms into the minimum set of transformations expressed in terms of the source table columns. Next, the software pushes all possible operations on tables of the same database down to that DBMS.

For example, the following data flow extracts rows from a single source table.

The first query selects only the rows in the source where column \( A \) contains a value greater than 100. The second query refines the extraction further, reducing the number of columns returned and further reducing the qualifying rows.
The software collapses the two queries into a single command for the DBMS to execute. The following command uses AND to combine the WHERE clauses from the two queries:

```
SELECT A, MAX(B), C
FROM source
WHERE A > 100 AND B = C
GROUP BY A, C
```

The software can push down all the operations in this SELECT statement to the source DBMS.

**Full push down from the source to the target example**

If the source and target are in the same datastore, the software can do a full push-down operation where the INSERT into the target uses a SELECT from the source. In the sample data flow in scenario 1, a full push down passes the following statement to the database:

```
INSERT INTO target (A, B, C)
SELECT A, MAX(B), C
FROM source
WHERE A > 100 AND B = C
GROUP BY A, C
```

If the source and target are not in the same datastore, the software can also do a full push-down operation if you use one of the following features:

- Add a Data_Transfer transform before the target.
- Define a database link between the two datastores.

**Full push down for auto correct load to the target example**

For supported databases, if you enable the **Auto correct load** and **Allow Merge** options, the Optimizer may be able to do a full push-down operation where the SQL statement is a MERGE into the target with a SELECT from the source.

In order for the **Allow Merge** option to generate a MERGE statement, the primary key of the source table must be a subset of the primary key of the
target table and the source row must be unique on the target key. In other words, there cannot be duplicate rows in the source data. If this condition is not met, the Optimizer pushes down the operation using a database-specific method to identify, update, and insert rows into the target table.

For example, in the following data flow, the source and target tables are in the same datastore and the Auto correct load and Allow Merge options are set to Yes.

The push-down operation passes the following statement to an Oracle database:

```
MERGE INTO "ODS"."TARGET" s
USING
SELECT "SOURCE"."A" A , "SOURCE"."B" B , "SOURCE"."C" C
FROM "ODS"."SOURCE"
) n
ON ((s.A = n.A))
WHEN MATCHED THEN
UPDATE SET s."B" = n.B,
  s."C" = n.C
WHEN NOT MATCHED THEN
INSERT /*+ APPEND */ (s."A", s."B", s."C" )
```

Similar statements are used for other supported databases.

### Partial push down to the source example

If the data flow contains operations that cannot be passed to the DBMS, the software optimizes the transformation differently than the previous two scenarios. For example, if Query1 called \( \text{func}(A) > 100 \), where \( \text{func} \) is a SAP BusinessObjects Data Services custom function, then the software generates two commands:
The first query becomes the following command which the source DBMS executes:

```sql
SELECT A, B, C
FROM source
WHERE B = C
```

The second query becomes the following command which SAP BusinessObjects Data Services executes because `func` cannot be pushed to the database:

```sql
SELECT A, MAX(B), C
FROM Query1
WHERE func(A) > 100
GROUP BY A, C
```

To view SQL

Before running a job, you can view the SQL code that SAP BusinessObjects Data Services generates for table sources in data flows. By examining the SQL code, you can verify that the software generates the commands you expect. If necessary, you can alter your design to improve the data flow.

1. Validate and save data flows.
2. Open a data flow in the workspace.
3. Select **Display Optimized SQL** from the **Validation** menu.

   Alternately, you can right-click a data flow in the object library and select **Display Optimized SQL**.

   The Optimize SQL window opens and shows a list of datastores and the optimized SQL code for the selected datastore. By default, the Optimize SQL window selects the first datastore, as the following example shows:
The software only shows the SELECT generated for table sources and INSERT INTO... SELECT for targets. It does not show the SQL generated for SQL sources that are not table sources, such as:

- Lookup function
- Key_generation function
- Key_Generation transform
- Table_Comparison transform

4. Select a name from the list of datastores on the left to view the SQL that this data flow applies against the corresponding database or application.

The following example shows the optimized SQL for the second datastore which illustrates a full push-down operation (INSERT INTO... SELECT). This data flows uses a Data_Transfer transform to create a transfer table that the software loads directly into the target. For more information, see "Data_Transfer transform for push-down operations" on page 1667.
In the Optimized SQL window you can:

- Use the **Find** button to perform a search on the SQL displayed.
- Use the **Save As** button to save the text as a .sql file.

If you try to use the **Display Optimized SQL** command when there are no SQL sources in your data flow, the software alerts you. Examples of non-SQL sources include:

- Message sources
- File sources
- IDoc sources

If a data flow is not valid when you click the **Display Optimized SQL** option, the software alerts you.

**Note:**
The Optimized SQL window displays the existing SQL statement in the repository. If you changed your data flow, save it so that the Optimized SQL window displays your current SQL statement.
Data_Transfer transform for push-down operations

Use the Data_Transfer transform to move data from a source or from another transform into the target datastore and enable a full push-down operation (INSERT INTO... SELECT) to the target. You can use the Data_Transfer transform to push-down resource-intensive operations that occur anywhere within a data flow to the database. Resource-intensive operations include joins, GROUP BY, ORDER BY, and DISTINCT.

Push down an operation after a blocking operation example

You can place a Data_Transfer transform after a blocking operation to enable SAP BusinessObjects Data Services to push down a subsequent operation. A blocking operation is an operation that the software cannot push down to the database, and prevents ("blocks") operations after it from being pushed down.

For example, you might have a data flow that groups sales order records by country and region, and sums the sales amounts to find which regions are generating the most revenue. The following diagram shows that the data flow contains a Pivot transform to obtain orders by Customer ID, a Query transform that contains a lookup_ext function to obtain sales subtotals, and another Query transform to group the results by country and region.

Because the Pivot transform and the lookup_ext function are before the query with the GROUP BY clause, the software cannot push down the
GROUP BY operation. The following Optimize SQL window shows the SELECT statement that the software pushes down to the source database:

```sql
SELECT "ORDERID", "CUSTOMERID", "EMPLOYEEID", "ORDERDATE", "REQUERDATE", "SHIPPEDDATE", "SHIPPEDBY", "FREIGHT", "SHEADDRESS", "SHIPCITY", "SHIPREGION", "SHIPPOSTALCODE", "SHIPCOUNTRY"
FROM "DB0"."ORDERS"
```

However, if you add a Data_Transfer transform before the second Query transform and specify a transfer table in the same datastore as the target table, the software can push down the GROUP BY operation.

The following Data_Transfer Editor window shows that the transfer type is table and the transfer table is in the same datastore as the target table.
The following Optimize SQL window shows that the software pushed down the GROUP BY to the transfer table TRANS2.
Using Data_Transfer tables to speed up auto correct loads example

Auto correct loading ensures that the same row is not duplicated in a target table, which is useful for data recovery operations. However, an auto correct load prevents a full push-down operation from the source to the target when the source and target are in different datastores.

For large loads using database targets that support the Allow Merge option for auto correct load, you can add a Data_Transfer transform before the target to enable a full push-down from the source to the target. In order for the Allow Merge option to generate a MERGE statement, the primary key of the source table must be a subset of the primary key of the target table and the source row must be unique on the target key. In other words, there cannot be duplicate rows in the source data. If this condition is not met, the Optimizer pushes down the operation using a database-specific method to identify, update, and insert rows into the target table.

Related Topics
- Operations that cannot be pushed down on page 44
- Reference Guide: Transforms, Data_Transfer
If the MERGE statement can be used, SAP BusinessObjects Data Services generates an SQL MERGE INTO `target` statement that implements the **Ignore columns with value** value (if a value is specified in the target transform editor) and the **Ignore columns with null** Yes/No setting.

For example, the following data flow loads sales orders into an Oracle target table which is in a different datastore from the source.

The following target editor shows that the **Auto correct load** option is activated (set to Yes). The **Ignore columns with null** and **Allow merge** options are also activated in this example.
The following Optimize SQL window shows the SELECT statement that the software pushes down to the source database.
When you add a Data_Transfer transform before the target and specify a transfer table in the same datastore as the target, the software can push down the auto correct load operation.

```sql
SELECT "ODS SALESORDER"."SALES_ORDER_NUMBER", "ODS SALESORDER"."ORDER_DATE", "ODS SALESORDER"."CUST_ID"
FROM "ODS"."ODS SALESORDER" "ODS SALESORDER"
```
The following Optimize SQL window shows the MERGE statement that the software will push down to the Oracle target.
Similar statements are used for other supported databases.

**Database link support for push-down operations across datastores**

Various database vendors support one-way communication paths from one database server to another. SAP BusinessObjects Data Services refers to communication paths between databases as database links. The datastores in a database link relationship are called linked datastores.

The software uses linked datastores to enhance its performance by pushing down operations to a target database using a target datastore. Pushing down operations to a database not only reduces the amount of information that needs to be transferred between the databases and SAP BusinessObjects Data Services but also allows the software to take advantage of the various DBMS capabilities, such as various join algorithms.

With support for database links, the software pushes processing down from different datastores, which can also refer to the same or different database type. Linked datastores allow a one-way path for data. For example, if you import a database link from target database B and link datastore B to datastore A, the software pushes the load operation down to database B, not to database A.

This section contains the following topics:
Software support

SAP BusinessObjects Data Services supports push-down operations using linked datastores on all Windows and Unix platforms. It supports DB2, Oracle, and MS SQL server databases.

To take advantage of linked datastores

1. Create a database link on a database server that you intend to use as a target in a job.

   The following database software is required. See the Supported Platforms document for specific version numbers.

   • For DB2, use the DB2 Information Services (previously known as Relational Connect) software and make sure that the database user has privileges to create and drop a nickname.

   To end users and client applications, data sources appear as a single collective database in DB2. Users and applications interface with the database managed by the information server. Therefore, configure an information server and then add the external data sources. DB2 uses nicknames to identify remote tables and views.

   See the DB2 database manuals for more information about how to create links for DB2 and non-DB2 servers.

   • For Oracle, use the Transparent Gateway for DB2 and MS SQL Server.

   See the Oracle database manuals for more information about how to create database links for Oracle and non-Oracle servers.
• For MS SQL Server, no special software is required.

Microsoft SQL Server supports access to distributed data stored in multiple instances of SQL Server and heterogeneous data stored in various relational and non-relational data sources using an OLE database provider. SQL Server supports access to distributed or heterogeneous database sources in Transact-SQL statements by qualifying the data sources with the names of the linked server where the data sources exist.

See the MS SQL Server database manuals for more information.

2. Create a database datastore connection to your target database.

Example of push-down with linked datastores

Linked datastores enable a full push-down operation (INSERT INTO...SELECT) to the target if all the sources are linked with the target. The sources and target can be in datastores that use the same database type or different database types.

The following diagram shows an example of a data flow that will take advantage of linked datastores:

The dataflow joins three source tables from different database types:
• ora_source.HRUSER1.EMPLOYEE on \oracle_server1
• ora_source_2.HRUSER2.PERSONNEL on \oracle_server2
• ora_source_3.HRUSER3.DEPARTMENT on \thinsqlserver
The software loads the join result into the target table ora_target.HRUSER3.EMP_JOIN on \oracle_server1.

In this data flow, the user (HRUSER3) created the following database links in the Oracle database oracle_server1.

<table>
<thead>
<tr>
<th>Database Link Name</th>
<th>Local (to database link location) Connection Name</th>
<th>Remote (to database link location) Connection Name</th>
<th>Remote User</th>
</tr>
</thead>
<tbody>
<tr>
<td>orasvr2</td>
<td>oracle_server1</td>
<td>oracle_server2</td>
<td>HRUSER2</td>
</tr>
<tr>
<td>tg4mysql</td>
<td>oracle_server1</td>
<td>mssql_server</td>
<td>DBO</td>
</tr>
</tbody>
</table>

To enable a full push-down operation, database links must exist from the target database to all source databases and links must exist between the following datastores:
- ora_target and ora_source
- ora_target and ora_source2
- ora_target and mssql_source

The software executes this data flow query as one SQL statement in oracle_server1:

```sql
INSERT INTO HR_USER3.EMP_JOIN (FNAME, ENAME, DEPTNO, SAL, COMM)
SELECT psnl.FNAME, emp.ENAME, dept.DEPTNO, emp.SAL, emp.COMM
FROM HR_USER1.EMPLOYEE emp, HR_USER2.PERSONNEL@orasvr2 psnl,
oracle_server1.mssql_server.DBO.DEPARTMENT@tg4msql dept;
```
Generated SQL statements

To see how SAP BusinessObjects Data Services optimizes SQL statements, use **Display Optimized SQL** from the **Validation** menu when a data flow is open in the workspace.

- For DB2, it uses nicknames to refer to remote table references in the SQL display.
- For Oracle, it uses the following syntax to refer to remote table references: `<remote_table>@<dblink_name>`.
- For SQL Server, it uses the following syntax to refer to remote table references: `<liked_server>.<remote_database>.<remote_user>.<remote_table>`.

Tuning performance at the data flow or Job Server level

You might want to turn off linked-datastore push downs in cases where you do not notice performance improvements.

For example, the underlying database might not process operations from different data sources well. Data Services pushes down Oracle stored procedures and external functions. If these are in a job that uses database links, it will not impact expected performance gains. However, Data Services does not push down functions imported from other databases (such as DB2). In this case, although you may be using database links, Data Services cannot push the processing down.

Test your assumptions about individual job designs before committing to a large development effort using database links.

For a data flow

On the data flow properties dialog, this product enables the **Use datastore links** option by default to allow push-down operations using linked datastores. If you do not want to use linked datastores in a data flow to push down processing, deselect the check box.
This product can perform push downs using datastore links if the tables involved share the same database type and database connection name, or datasource name, even if the tables have different schema names. However, problems with enabling this feature could arise, for example, if the user of one datastore does not have access privileges to the tables of another datastore, causing a data access problem. In such a case, you can disable this feature.

For a Job Server

You can also disable linked datastores at the Job Server level. However, the Use database links option, at the data flow level, takes precedence.

Related Topics
• Designer Guide: Executing Jobs, Changing Job Server options
Using Caches
Caching data

You can improve the performance of data transformations that occur in memory by caching as much data as possible. By caching data, you limit the number of times the system must access the database.

SAP BusinessObjects Data Services provides the following types of caches that your data flow can use for all of the operations it contains:

- **In-memory**
  
  Use in-memory cache when your data flow processes a small amount of data that fits in memory.

- **Pageable cache**
  
  Use pageable cache when your data flow processes a very large amount of data that does not fit in memory. When memory-intensive operations (such as Group By and Order By) exceed available memory, the software uses pageable cache to complete the operation.

Pageable cache is the default cache type. To change the cache type, use the **Cache type** option on the data flow Properties window.

**Note:**

If your data fits in memory, it is recommended that you use in-memory cache because pageable cache incurs an overhead cost.

This section includes the following topics:

- Caching sources
  - **Caching sources** on page 67
  - **Caching joins** on page 68
- Caching intermediate data
  - **Caching lookups** on page 69
• **Caching table comparisons** on page 70
• **Specifying a pageable cache directory** on page 71 for memory-intensive operations

## Caching sources

By default, the Cache option is set to *Yes* in a source table or file editor to specify that data from the source is cached using memory on the Job Server computer. The default value for Cache type for data flows is *Pageable*.

It is recommended that you cache small tables in memory. Calculate the approximate size of a table with the following formula to determine if you should use a cache type of *Pageable* or *In-memory*.

\[
\text{table size} = (\text{in bytes}) = \text{# of rows} \times \text{# of columns} \times 20 \text{ bytes (average column size)} \times 1.3 \text{ (30% overhead)}
\]

Compute row count and table size on a regular basis, especially when:
• You are aware that a table has significantly changed in size.
• You experience decreased system performance.

If the table fits in memory, change the value of **Cache type** to *In-memory* in the Properties window of the data flow.
Caching joins

Cache a source only if it is being used as an inner source (or "inner loop"). (Inner sources or loops have a lower join rank than outer sources or loops). Caching does not affect the order in which tables are joined. If optimization conditions are such that the software is pushing down operations to the underlaying database, it ignores your cache setting.

If a table becomes too large to fit in the cache, ensure that the cache type is pageable.

To change the cache type for a data flow

1. In the object library, select the data flow name.
2. Right-click and choose Properties.
3. On the General tab of the Properties window, select **Pageable** in the drop-down list for the **Cache type** option.

**Caching lookups**

You can also improve performance by caching data when looking up individual values from tables and files.

There are two methods of looking up data:

- **Using a Lookup function in a query** on page 69
- **Using a source table and setting it as the outer join** on page 70

**Using a Lookup function in a query**

SAP BusinessObjects Data Services has three Lookup functions: **lookup**, **lookup_seq**, and **lookup_ext**. The **lookup** and **lookup_ext** functions have cache options. Caching lookup sources improves performance because the software avoids the expensive task of creating a database query or full file scan on each row.

You can set cache options when you specify a lookup function. There are three caching options:

- **NO_CACHE** — Does not cache any values.
- **PRE_LOAD_CACHE** — Preloads the result column and compare column into memory (it loads the values before executing the lookup).
- **DEMAND_LOAD_CACHE** — Loads the result column and compare column into memory as the function executes.

Use this option when looking up highly repetitive values that are a small subset of the data and when missing values are unlikely.

Demand-load caching of lookup values is helpful when the lookup result is the same value multiple times. Each time the software cannot find the value in the cache, it must make a new request to the database for that value. Even if the value is invalid, the software has no way of knowing if it is missing or just has not been cached yet.
When there are many values and some values might be missing, demand-load caching is significantly less efficient than caching the entire source.

**Using a source table and setting it as the outer join**

Although you can use lookup functions inside SAP BusinessObjects Data Services queries, an alternative is to expose the translate (lookup) table as a source table in the data flow diagram, and use an outer join (if necessary) in the query to look up the required data. This technique has some advantages:

- You can graphically see the table the job will search on the diagram, making the data flow easier to maintain
- The software can push the execution of the join down to the underlying RDBMS (even if you need an outer join)

This technique also has some disadvantages:

- You cannot specify default values in an outer join (default is always null), but you can specify a default value in lookup_ext.
- If an outer join returns multiple rows, you cannot specify what to return, (you can specify MIN or MAX in lookup_ext).
- The workspace can become cluttered if there are too many objects in the data flow.
- There is no option to use DEMAND_LOAD caching, which is useful when looking up only a few repetitive values in a very large table.

**Tip:**
If you use the lookup table in multiple jobs, you can create a persistent cache that multiple data flows can access. For more information, see *Using persistent cache* on page 72.

**Caching table comparisons**

You can improve the performance of a Table_Comparison transform by caching the comparison table. There are three modes of comparisons:
• Row-by-row select
• Cached comparison table
• Sorted input

Of the three, **Row-by-row select** will likely be the slowest and **Sorted input** the fastest.

**Tip:**
• If you want to sort the input to the table comparison transform, then choose the **Sorted input** option for comparison.
• If the input is not sorted, then choose the **Cached comparison table** option.

### Specifying a pageable cache directory

If the memory-consuming operations in your data flow exceed the available memory, SAP Business Objects Data Services uses pageable cache to complete the operation. Memory-intensive operations include the following operations:

• Distinct
• Functions such as count_distinct and lookup_ext
• Group By
• Hierarchy_Flattening
• Order By

**Note:**
The default pageable cache directory is `%LINKDIR\Log\PCache`. If your data flows contain memory-consuming operations, change this value to a pageable cache directory that:

• Contains enough disk space for the amount of data you plan to profile.
• Is on a separate disk or file system from the SAP BusinessObjects Data Services system.

Change the directory in the **Specify a directory with enough disk space for pageable cache** option in the Server Manager, under Runtime resources configured for this computer.
Using persistent cache

Persistent cache datastores provide the following benefits for data flows that process large volumes of data.

- You can store a large amount of data in persistent cache which SAP BusinessObjects Data Services quickly pages into memory each time the job executes. For example, you can access a lookup table or comparison table locally (instead of reading from a remote database).

- You can create cache tables that multiple data flows can share (unlike a memory table which cannot be shared between different real-time jobs). For example, if a large lookup table used in a lookup_ext function rarely changes, you can create a cache once and subsequent jobs can use this cache instead of creating it each time.

Persistent cache tables can cache data from relational database tables and files.

**Note:**
You cannot cache data from hierarchical data files such as XML messages and SAP IDocs (both of which contain nested schemas). You cannot perform incremental inserts, deletes, or updates on a persistent cache table.

You create a persistent cache table by loading data into the persistent cache target table using one data flow. You can then subsequently read from the cache table in another data flow. When you load data into a persistent cache table, SAP BusinessObjects Data Services always truncates and recreates the table.

Using persistent cache tables as sources

After you create a persistent cache table as a target in one data flow, you can use the persistent cache table as a source in any data flow. You can also use it as a lookup table or comparison table.

**Related Topics**
- Reference Guide: Objects, Persistent cache source
Monitoring and tuning caches

This section describes the following topics:

Related Topics
• Using statistics for cache self-tuning on page 73
• To monitor and tune in-memory and pageable caches on page 73

Using statistics for cache self-tuning

SAP BusinessObjects Data Services uses cache statistics collected from previous job runs to automatically determine which cache type to use for a data flow. Cache statistics include the number of rows processed.

The default cache type is pageable. the software can switch to in-memory cache when it determines that your data flow processes a small amount of data that fits in memory.

To automatically choose the cache type

1. Run your job with options Collect statistics for optimization.
2. Run your job again with option Use collected statistics (this option is selected by default).

To monitor and tune in-memory and pageable caches

You can also monitor and choose the cache type to use for the data flow.

1. Test run your job with options Collect statistics for optimization and Collect statistics for monitoring.

   Note:
   The option Collect statistics for monitoring is very costly to run because it determines the cache size for each row processed.

2. Run your job again with option Use collected statistics (this option is selected by default).
3. Look in the Trace Log to determine which cache type was used.
   • The first time you run the job or if you have not previously collected statistics, the following messages indicate that cache statistics are not available and the sub data flows use the default cache type, pageable.

   Cache statistics for sub data flow <GroupBy_DF_1_1> are not available to be used for optimization and need to be collected before they can be used.

   Sub data flow <GroupBy_DF_1_1> using PAGEABLE Cache with <1280 MB> buffer pool.

   • You might see the following message that indicates that the software is switching to In-memory cache:

   Cache statistics determined that sub data flow <GroupBy_DOP2_DF_1_4> uses <1> caches with a total size of <1920> bytes. This is less than (or equal to) the virtual memory <1342177280> bytes available for caches. Statistics is switching the cache type to IN MEMORY.

   Sub data flow <GroupBy_DOP2_DF_1_4> using IN MEMORY Cache.

   Because pageable cache is the default cache type for a data flow, you might want to permanently change Cache type to In-Memory in the data flow Properties window.
• You might see the following messages that indicate on sub data flow uses IN MEMORY cache and the other sub data flow uses PAGEABLE cache:

Sub data flow <Orders_Group_DF_1> using IN MEMORY Cache.

...

Sub data flow <Orders_Group_DF_2> using PAGEABLE Cache with <1536 MB> buffer pool.

4. Look in the Administrator Performance Monitor to view data flow statistics and see the cache size.
   a. On the Administrator, select Batch > repository
   b. On the Batch Job Statuspage, find a job execution instance.
   c. Under Job Information for an instance, click Performance > Monitor.
The Administrator opens the Graph tab of the Performance Monitor page. This tab shows a graphical view of the start time, stop time, and execution time for each work flow, data flow, and sub data flow within the job.

d. To display the data flow times in a tabular format, click the Table tab.

e. To display statistics for each object within a data flow or sub data flow, click either the corresponding execution bar on the Graph tab, or one of the data flow names on the Table tab. The Transform tab displays the following statistics.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name that you gave the object (source, transform, or target) in the Designer.</td>
</tr>
<tr>
<td>Statistic</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Type</td>
<td>Type of object within the data flow. Possible values include Source, Mapping, Target.</td>
</tr>
<tr>
<td>Start time</td>
<td>Date and time this object instance started execution.</td>
</tr>
<tr>
<td>End time</td>
<td>Date and time this object instance stopped execution.</td>
</tr>
<tr>
<td>Execution time (sec)</td>
<td>Time (in seconds) the object took to complete execution.</td>
</tr>
<tr>
<td>Row Count</td>
<td>Number of rows that this object processed.</td>
</tr>
<tr>
<td>Cache Size (KB)</td>
<td>Size (in kilobytes) of the cache that was used to process this object.</td>
</tr>
</tbody>
</table>

**Note:**
This statistics displays only if you selected **Collect statistics for monitoring** for the job execution.

For example, click the top execution bar or the name Group_Orders_DF to display the following statistics for both sub data flows.
5. If the value in Cache Size is approaching the physical memory limit on the job server, consider changing the **Cache type** of a data flow from **In-memory** to **Pageable**.
Using Parallel Execution
You can set SAP BusinessObjects Data Services to perform data extraction, transformation, and loads in parallel by setting parallel options for sources, transforms, and targets. In addition, you can set individual data flows and work flows to run in parallel by simply not connecting them in the workspace. If the Job Server is running on a multi-processor computer, it takes full advantage of available CPUs.

This section contains the following topics:
- Parallel data flows and work flows on page 80
- Parallel execution in data flows on page 81

**Parallel data flows and work flows**

You can explicitly execute different data flows and work flows in parallel by not connecting them in a work flow or job. SAP BusinessObjects Data Services coordinates the parallel steps, then waits for all steps to complete before starting the next sequential step.

For example, use parallel processing to load dimension tables by calling work flows in parallel. Then specify that your job creates dimension tables before the fact table by moving it to the left of a second (parent) work flow and connecting the flows.

Parallel engine processes execute the parallel data flow processes. Note that if you have more than eight CPUs on your Job Server computer, you can increase **Maximum number of engine processes** to improve
performance. To change the maximum number of parallel engine processes, use the Job Server options (Tools > Options > Job Server > Environment).

Parallel execution in data flows

For batch jobs, SAP BusinessObjects Data Services allows you to execute parallel threads in data flows.

This section contains the following:

• Table partitioning on page 81
• Degree of parallelism on page 87
• Combining table partitioning and a degree of parallelism on page 95
• File multi-threading on page 98

Table partitioning

SAP BusinessObjects Data Services processes data flows with partitioned tables based on the amount of partitioning defined. There are three basic scenarios:

• Data flow with source partitions only on page 81
• Data flow with target partitions only on page 82
• Dataflow with source and target partitions on page 82

Data flow with source partitions only

If you have a data flow with a source that has two partitions connected to a query and a target, it appears in the workspace as shown in the following diagram:

At runtime, the software translates this data flow to:
The software instantiates a source thread for each partition, and these threads run in parallel. The data from these threads later merges into a single stream by an internal merge transform before processing the query.

**Data flow with target partitions only**

If you have a data flow with a target that has two partitions connected to a query and a source, it appears in the workspace as shown in the following diagram:

At runtime, the software translates this data flow to:

The software inserts an internal Round Robin Split (RRS) transform after the Query transform, which routes incoming rows in a round-robin fashion to internal Case transforms. The Case transforms evaluate the rows to determine the partition ranges. Finally, an internal Merge transform collects the incoming rows from different Case transforms and outputs a single stream of rows to the target threads. The Case, Merge, and the target threads execute in parallel.

**Dataflow with source and target partitions**

If you have a data flow with a source that has two partitions connected to a query and a target that has two partitions, it appears in the workspace as shown in the following diagram:
At runtime, the software translates this data flow to:

The source threads execute in parallel and the Case, Merge, and targets execute in parallel.

**Viewing, creating, and enabling table partitions**

Oracle databases support range, list, and hash partitioning. You can import this information as table metadata and use it to extract data in parallel. You can use range and list partitions to load data to Oracle targets. You can also specify logical range and list partitions using SAP BusinessObjects Data Services metadata for Oracle tables.

In addition, it provides the ability to specify logical range partitions for DB2, Microsoft SQL Server, Sybase ASE, and Sybase IQ tables by modifying imported table metadata.

SAP BusinessObjects Data Services uses partition information by instantiating a thread at runtime for each partition. These threads execute in parallel. To maximize performance benefits, use a multi-processor environment.

**To view partition information**

1. Import a table into SAP BusinessObjects Data Services.
2. In the **Datastores** tab of the object library, right-click the table name and select **Properties**.
3. Click the **Partitions** tab.

When you import partitioned tables from your database, you will find these partitions displayed on the Partitions tab of the table's Properties window. The partition name appears in the first column. The columns that are used for partitioning appear as column headings in the second row.
If you import a table that does not have partitions, you can create logical partitions using the Partitions tab of the table's Properties window.

To create or edit table partition information

1. In the Datastores tab of the object library, right-click the table name and select Properties.
2. In the Properties window, click the Partitions tab.
3. Select a partition type.

<table>
<thead>
<tr>
<th>Partition Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>This table is not partitioned.</td>
</tr>
<tr>
<td>Range</td>
<td>Each partition contains a set of rows with column values less than those specified. For example, if the value of column one is 100,000, then the data set for partition one will include rows with values less than 100,000 in column one.</td>
</tr>
<tr>
<td>List</td>
<td>Each partition contains a set of rows that contain the specified column values.</td>
</tr>
</tbody>
</table>

Note: If you imported an Oracle table with hash partitions, you cannot edit the hash settings in SAP BusinessObjects Data Services. The Partitions tab
displays the hash partition name and ID as read-only information. However, you can change the partition type to Range or List to create logical range or list partitions for an Oracle table imported with hash partitions.

4. Add, insert, or remove partitions and columns using the tool bar. (See table at the end of this procedure.)

5. Select the name of a column from each column list box.

6. Enter column values for each partition.

SAP BusinessObjects Data Services validates the column values entered for each partition according to the following rules:

- Values can be literal numbers and strings or datetime types.
- Column values must match column data types.
- Literal strings must include single quotes: 'Director'.
- For range partitions, the values for a partition must be greater than the values for the previous partition.
- For the last partition, you can enter the value MAXVALUE to include all values.
7. Click **OK**.

If the validation rules described in the previous step are not met, you will see an error message.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add Partition" /></td>
<td>Add Partition</td>
</tr>
<tr>
<td><img src="image" alt="Insert Partition" /></td>
<td>Insert Partition</td>
</tr>
<tr>
<td><img src="image" alt="Remove Partition" /></td>
<td>Remove Partition</td>
</tr>
<tr>
<td><img src="image" alt="Add Column" /></td>
<td>Add Column</td>
</tr>
<tr>
<td><img src="image" alt="Insert Column" /></td>
<td>Insert Column</td>
</tr>
<tr>
<td><img src="image" alt="Remove Column" /></td>
<td>Remove Column</td>
</tr>
</tbody>
</table>

The number of partitions in a table equals the maximum number of parallel instances that the software can process for a source or target created from this table.

In addition to importing partitions or creating and editing partition metadata, enable the partition settings when you configure sources and targets.

**To enable partition settings in a source or target table**

1. Drop a table into a data flow and select **Make Source** or **Make Target**.
2. Click the name of the table to open the source or target table editor.
3. Enable partitioning for the source or target:
   a. For a source table, click the **Enable Partitioning** check box.
   b. For a target table, click the **Options** tab, then click the **Enable Partitioning** check box.
4. Click OK.

When the job executes, the software generates parallel instances based on the partition information.

**Note:**
If you are loading to partitioned tables, a job will execute the load in parallel according to the number of partitions in the table. If you set **Enable Partitioning** to Yes and **Include in transaction** to No, the **Include in transaction** setting overrides the **Enable Partitioning** option. For example, if your job is designed to load to a partitioned table but you set **Include in transaction** to Yes and enter a value for **Transaction order**, when the job executes, the software will include the table in a transaction load and does not parallel load to the partitioned table.

**Tip**
If the underlying database does not support range partitioning and if you are aware of a natural distribution of ranges, for example using an Employee Key column in an Employee table, then you can edit the imported table metadata and define table ranges. The software would then instantiate multiple reader threads, one for each defined range, and execute them in parallel to extract the data.

**Note:**
Table metadata editing for partitioning is designed for source tables. If you use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in SAP BusinessObjects Data Services. If there is a mismatch, the software will not use the partition name to load partitions. Consequently, the whole table updates.

**Degree of parallelism**

Degree Of Parallelism (DOP) is a property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. If there are multiple transforms in a data flow, SAP BusinessObjects Data Services chains them together until it reaches a merge point.

You can run transforms in parallel by entering a number in the **Degree of Parallelism** box on a data flow's Properties window. The number is used to
replicate transforms in the data flow which run as separate threads when the Job Server processes the data flow.

This section describes the following parallel operations:

- **Degree of parallelism and transforms** on page 88
- **Degree of parallelism and joins** on page 91
- **Degree of parallelism and functions** on page 93

**Degree of parallelism and transforms**

The Query transform always replicates when you set DOP to a value greater than 1. SAP BusinessObjects Data Services also replicates query operations such as Order By, Group By, join, and functions such as lookup_ext.

The Table Comparison replicates when you use the Row-by-row select and Cached comparison table comparison methods.

- Map_Operation
- History_Preserving
- Pivot

There are two basic scenarios:

- DOP and a data flow with a single transform
- DOP and a data flow with multiple transforms

**DOP and a data flow with a single transform**

The following figures show runtime instances of a data flow with a DOP of 1, and the same data flow with a DOP of 2.

![Figure 7-1: Runtime instance of a data flow where DOP =1](image)
With a DOP greater than 1, the software inserts an internal Round Robin Split (RRS) that transfers data to each of the replicated queries. The replicated queries execute in parallel, and the results merge into a single stream by an internal Merge transform.

**DOP and a data flow with multiple transforms**

The following figures show runtime instances of a data flow with a DOP of 1, and the same data flow with a DOP of 2. Notice multiple transforms in a data flow replicate and chain when the DOP is greater than 1.

When there are multiple transforms in a data flow and the DOP is greater than 1, the software carries the replicated stream as far as possible, then merges the data into a single stream.

**To set the Degree of Parallelism for a data flow**

The degree of parallelism (DOP) is a data flow property that acts on transforms added to the data flow.

1. In the object library, select the Data Flow tab.
2. Right-click the data flow icon and select **Properties**.
3. Enter a number for **Degree of parallelism**.

![Properties dialog box](image)

The default value for degree of parallelism is 0. If you set an individual data flow’s degree of parallelism to this default value, then you can control it using a Global_DOP value which affects all data flows run by a given Job Server. If you use any other value for a data flow's degree of parallelism, it overrides the Global_DOP value.

You can use the local and global DOP options in different ways. For example:

- If you want to globally set all data flow DOP values to 4, but one data flow is too complex and you do not want it to run in parallel, you can set the **Degree of parallelism** for this data flow locally. From the data flow’s Properties window, set this data flow’s **Degree of parallelism** to 1. All other data flows will replicate and run transforms in parallel after you set the Global_DOP value to 4. The default for the Global_DOP value is 1.
• If you want to set the DOP on a case-by-case basis for each data flow, set the value for each data flow's **Degree of parallelism** to any value except zero. You set the Global_DOP value in the Job Server options.

4. Click **OK**.

**Related Topics**

• *Designer Guide: Executing Jobs, Changing Job Server options*

---

**Degree of parallelism and joins**

If your Query transform joins sources, DOP determines the number of times the join replicates to process a parallel subset of data.

This section describes two scenarios:

• DOP and executing a join as a single process
• DOP and executing a join as multiple processes

**DOP and executing a join as a single process**

The following figures show runtime instances of a data flow that contains a join with a DOP of 1 and the same data flow with a DOP of 2. You use join ranks to define the outer source and inner source (see *Join ordering* on page 181. In both data flows, the inner source is cached in memory (see *Caching joins* on page 68.)

![Figure 7-5: Runtime instance of a join where DOP =1](image)

*Figure 7-5: Runtime instance of a join where DOP =1*
With a DOP greater than one, the software inserts an internal Round Robin Split (RRS) that transfers data to each of the replicated joins. The inner source is cached once, and each half of the outer source joins with the cached data in the replicated joins. The replicated joins execute in parallel, and the results merge into a single stream by an internal Merge transform.

**DOP and executing a join as multiple processes**

When you select the Run JOIN as a separate process in the Query transform, you can split the execution of a join among multiple processes. The software creates a sub data flow for each separate process.

The following figure shows a runtime instance of a data flow that contains a join with a DOP of 2 and the Run JOIN as a separate process option selected.

The data flow becomes four sub data flows (indicated by the blue dotted and dashed line in the figure):

- The first sub data flow uses an internal hash algorithm to split the data.
• The next two sub data flows are the replicated joins that run as separate processes.
• The last sub data flow merges the data and loads the target.

Tip:
If DOP is greater than one, select either job or data flow for the Distribution level option when you execute the job. If you execute the job with the value sub data flow for Distribution level, the Hash Split sends data to the replicated queries that might be executing on different Job Servers. Because the data is sent on the network between different Job Servers, the entire data flow might be slower. For more information about job distribution levels, see Using grid computing to distribute data flow execution on page 118.

Degree of parallelism and functions

You can set stored procedures and custom functions to replicate with the transforms in which they are used. To specify this option, select the Enable parallel execution check box on the function's Properties window. If this option is not selected and you add the function to a transform, the transform will not replicate and run in parallel even if its parent data flow has a value greater than 1 set for Degree of parallelism.

When enabling functions to run in parallel, verify that:
• Your database will allow a stored procedure to run in parallel
• A custom function set to run in parallel will improve performance

All built-in functions, except the following, replicate if the transform they are used in replicates due to the DOP value:

<table>
<thead>
<tr>
<th>avg()</th>
<th>min()</th>
</tr>
</thead>
<tbody>
<tr>
<td>count()</td>
<td>previous_row_value()</td>
</tr>
<tr>
<td>count_distinct()</td>
<td>print()</td>
</tr>
<tr>
<td>double_metaphone()</td>
<td>raise_exception()</td>
</tr>
</tbody>
</table>
To enable stored procedures to run in parallel

Use the **Enable parallel execution** option to set functions to run in parallel when the transforms in which they are used execute in parallel.

1. In the **Datastores** tab of the object library, expand a **Datastore** node.
2. Expand its **Function** node.
3. Right-click a function and select **Properties**.
4. In the Properties window, click the **Function** tab.
5. Click the **Enable Parallel Execution** check box.
6. Click **OK**.

To enable custom functions to run in parallel

1. In the **Custom Functions** tab of the object library, right-click a function name and select **Properties**.

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>exec()</code></td>
<td><code>raise_exception_ext()</code></td>
</tr>
<tr>
<td><code>get_domain_description()</code></td>
<td><code>set_env()</code></td>
</tr>
<tr>
<td><code>gen_row_num()</code></td>
<td><code>sleep()</code></td>
</tr>
<tr>
<td><code>gen_row_num_by_group()</code></td>
<td><code>smtp_to()</code></td>
</tr>
<tr>
<td><code>is_group_changed()</code></td>
<td><code>soundex()</code></td>
</tr>
<tr>
<td><code>key_generation()</code></td>
<td><code>sql()</code></td>
</tr>
<tr>
<td><code>mail_to()</code></td>
<td><code>sum()</code></td>
</tr>
<tr>
<td><code>max()</code></td>
<td><code>total_rows()</code></td>
</tr>
</tbody>
</table>
2. In the Properties window, click the **Function** tab.
3. Click the **Enable Parallel Execution** check box.
4. Click **OK**.

**Tips**

DOP can degrade performance if you do not use it judiciously. The best value to choose depends on the complexity of the flow and number of CPUs available. For example, on a computer with four CPUs, setting a DOP greater than two for the following data flow will not improve performance but can potentially degrade it due to thread contention.

![Diagram of data flow](image)

If your data flow contains an Order By or a Group By that is not pushed down to the database, put them at the end of a data flow. A sort node (Order By, Group By) is always a merge point, after which the engine proceeds as if the DOP value is 1. For information on viewing the SQL statements pushed down to the database, see *To view SQL* on page 48.

**Combining table partitioning and a degree of parallelism**

Different settings for source and target partitions and the degree of parallelism result in different behaviors in the SAP BusinessObjects Data Services engine. The sections that follow show some examples. For all the following scenarios, the data flow appears as follows:

![Diagram of data flow](image)
Two source partitions and a DOP of three

When a source has two partitions, it replicates twice. The input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP (in this case, three outputs to the query transform). The stream then merges and feeds into the target.

**Tip:**
If the target is not partitioned, set the **Number of loaders** option equal to the DOP value. Depending on the number of CPUs available, set the DOP value equal to the number of source partitions as a general rule. This produces a data flow without the Merge Round Robin Split and each partition pipes the data directly into the consuming transform.

Two source partitions and a DOP of two

When the number of source partitions is the same as the value for DOP, the engine merges before the target (or before any operation that requires a merge, such as aggregation operations) and proceeds in a single stream to complete the flow.
Two source partitions, DOP of three, two target partitions

When the number of source partitions is less than the value for DOP, the input feeds into a merge-round-robin splitter (MRRS) that merges the input streams and splits them into a number equal to the value for DOP. The engine then merges the data before the target to equal the number of target partitions, then proceeds to complete the flow.

Tip:
If the number of target partitions is not equal to the number of source partitions, set the **Number of loaders** option equal to the DOP value and do not enable partitioning for the target. Depending on the number of CPUs available, set the DOP value equal to the number of source partitions as a general rule. This produces a data flow without the Merge Round Robin Split and each partition pipes the data directly into the consuming transform.

Two source partitions, DOP of two, and two target partitions

The best case situation is when the following conditions exist:

- The source and target are partitioned the same way.
- The source and target have the same number of partitions.
- DOP is equal to the same number of partitions.

When a source has two partitions, it replicates twice. Because the DOP value is two, the query transform replicates twice. When a target has two partitions, it replicates twice. The following figure shows that each source partition feeds directly into a replicated query transform, and the output from each query feeds directly into a replicated target.
File multi-threading

You can set the number of threads used to process flat file sources and targets. The **Parallel process threads** option is available on the:

- File format editor
- Source file editor
- Target file editor
- Properties window of an ABAP data flow.

Without multi-threading:

- With delimited file reading, the SAP BusinessObjects Data Services reads a block of data from the file system and then scans each character to determine if it is a column delimiter, a row delimiter, or a text delimiter. Then it builds a row using an internal format.
- For positional file reading, the software does not scan character by character, but it still builds a row using an internal format.
- For file loading, processing involves building a character-based row from the internal row format.

You can set these time-consuming operations to run in parallel. You can use the **Parallel process threads** option to specify how many threads to execute in parallel to process the I/O blocks.

**Note:**
Enabling CPU hyperthreading can negatively affect the performance of servers and is therefore not supported.

**Related Topics**

- *Designer Guide: File Formats*
- *Reference Guide: Objects, Source*
- *Reference Guide: Objects, Target*
Flat file sources

To use the **Parallel process threads** option, the following conditions must be met:

- In the file format editor:
  - For delimited files, no text delimiters are defined. For fixed-width files, having a text delimiter defined does not prevent the file from being read by parallel process threads.
  
  You can set SAP BusinessObjects Data Services to read flat file data in parallel in most cases because the majority of jobs use fixed-width or column-delimited source files that do not have text delimiters specified.
  - An end-of-file (EOF) marker for the file's input/output style is not specified.
  - The value of the row delimiter is not set to `{none}`. A row delimiter can be `{none}` only if the file is a fixed-width file.
  - If the file has a multi-byte locale and you want to take advantage of parallel process threads, set the row delimiter as follows:
    - The length of the row delimiter must be 1. If the codepage of the file is UTF-16, the length of the row delimiter can be 2.
    - The row delimiter hex value must be less than 0x40.
  
  - In the Source File Editor, no number has been entered for **Rows to read**.
    
    The Rows to read option indicates the maximum number of rows that the software reads. It is normally used for debugging. Its default value is none.
  
  - The maximum row size does not exceed 128 KB.

If a file source needs to read more than one file, for example, *.txt is specified for the **File(s)** option in the file format editor, the software processes the data in the first file before the data in the next file. It performs file multi-threading one file at a time.
Flat file targets

If you enter a positive value for **Parallel process threads**, Data Services parallel processes flat file targets when the maximum row size does not exceed 128KB.

Tuning performance

The **Parallel process threads** option is a performance enhancement for flat file sources and targets. Performance is defined as the total elapsed time used to read a file source.

A multi-threaded file source or target achieves high performance for reads and loads by maximizing the utilization of the CPUs on your Job Server computer. You will notice higher CPU usage when you use this feature. You might also notice higher memory usage because the number of process threads you set (each consisting of blocks of rows that use 128 kilobytes) reside in memory at the same time.

To tune performance, adjust the value for **Parallel process threads**. Ideally, have at least as many CPUs as process threads. For example, if you enter the value 4 for **Parallel process threads**, have at least four CPUs on your Job Server computer.

However, increasing the value for process threads does not necessarily improve performance. The file reads and loads achieve their best performance when the work load is distributed evenly among all the CPUs and the speed of the file's input/output (I/O) thread is comparable with the speed of the process threads.

The I/O thread for a file source reads data from a file and feeds it to process threads. The I/O thread for a file target takes data from process threads and loads it to a file. Therefore, if a source file's I/O thread is too slow to keep the process threads busy, there is no need to increase the number of process threads.

If there is more than one process thread on one CPU, that CPU will need to switch between the threads. There is an overhead incurred in creating these threads and switching the CPU between them.
Tips

The best value for **Parallel process threads** depends on the complexity of your data flow and the number of available processes. If your Job Server is on a computer with multiple CPUs, the values for file sources and targets should be set to at least two.

After that, experiment with different values to determine the best value for your environment.

Here are some additional guidelines:

- If **Parallel process threads** is set to *none*, then flat file reads and loads are not processed in parallel.

- If **Parallel process threads** is set to 1, (meaning that one process thread will spawn) and your Job Server computer has one CPU, then reads and loads can occur faster than single-threaded file reads and loads because SAP BusinessObjects Data Services reads the I/O thread separately and concurrently with the process thread.

- If **Parallel process threads** is set to 4, four process threads will spawn. You can run these threads on a single CPU. However, using four CPUs would more likely maximize the performance of flat file reads and loads.
Distributing Data Flow Execution
The previous chapter describes how SAP BusinessObjects Data Services can run a single process as multiple threads that run in parallel on a multiprocessor computer. Using Degree of Parallelism (DOP), it can execute each thread on a separate CPU on the computer.

This section describes how the software can split a process (data flow) into multiple processes (sub data flows) that can take advantage of more memory across multiple computers or on the same computer that has more than two gigabytes of memory. For example, if your computer has eight gigabytes of memory, you can have four sub data flows that each can use up to two gigabytes.

With this capability, the software can distribute CPU-intensive and memory-intensive operations (such as join, grouping, table comparison and lookups). This distribution of data flow execution provides the following potential benefits:

- Better memory management by taking advantage of more CPU power and physical memory
- Better job performance and scalability by taking advantage of grid computing

You can create sub data flows so that the software does not need to process the entire data flow in memory at one time. You can also distribute the sub data flows to different job servers within a server group to use additional memory and CPU resources.

This section contains the following topics:

- Splitting a data flow into sub data flows on page 104
- Using grid computing to distribute data flow execution on page 118

**Splitting a data flow into sub data flows**

Use the following features to split a data flow into multiple sub data flows:

- Run as a separate process option on page 105
- Data_Transfer transform on page 112
Run as a separate process option

If your data flow contains multiple resource-intensive operations, you can run each operation as a separate process (sub data flow) that uses separate resources (memory and computer) from each other to improve performance and throughput. When you specify multiple Run as separate process options in objects in a data flow, SAP BusinessObjects Data Services splits the data flow into sub data flows that run in parallel.

The Run as a separate process option is available on resource-intensive operations that including the following:

- Hierarchy_Flattening transform
- Associate transform
- Country ID transform
- Global Address Cleanse transform
- Global Suggestion Lists transform
- Match Transform
- United States Regulatory Address Cleanse transform
- User-Defined transform
- Query operations that are CPU-intensive and memory-intensive:
  - Join
  - GROUP BY
  - ORDER BY
  - DISTINCT
- Table_Comparison transform
- Lookup_ext function
- Count_distinct function
- Search_replace function

Examples of multiple processes for a data flow

A data flow can contain multiple resource-intensive operations that each require large amounts of memory or CPU utilization. You can run each resource-intensive operation as a separate process that can use more memory on a different computer or on the same computer that has more than two gigabytes of memory. For example, you might have a data flow that
sums sales amounts from a lookup table and groups the sales by country and region to find which regions are generating the most revenue. The data flow contains a Query transform for the lookup_ext function to obtains sales subtotals and another Query transform to group the results by country and region.

To define separate processes in this sample data flow, take one of the following actions:

- When you define the Lookup_ext function in the first query transform, select the **Run as a separate process** option.
• When you define the Group By operation in the second query transform, select the **Run GROUP BY as a separate process** option on the **Advanced** tab.

<table>
<thead>
<tr>
<th>Mapping</th>
<th>SELECT</th>
<th>FROM</th>
<th>OUTER JOIN</th>
<th>WHERE</th>
<th>GROUP BY</th>
<th>ORDER BY</th>
<th>Search/Replace</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Scenario 1: Run multiple sub data flows with DOP set to 1**

The following diagram shows how SAP BusinessObjects Data Services splits this data flow into two sub data flows when you specify the **Run as a separate process** option for either the Lookup_ext function or the Group By.

The software generates sub data flow names that follow this format:

\[DFName_{executionGroupNumber}_{indexInExecutionGroup}\]

• **DFName** is the name of the data flow.
• **executionGroupNumber** is the order that the software executes a group of sub data flows
• **indexInExecutionGroup** is the sub data flow within an execution group.

When you execute the job, the Trace Log shows that the software creates two sub data flows that execute in parallel and have different process IDs (Pids). For example, the following trace log shows two sub data flows GroupBy_DF_1_1 and GroupBy_DF_1_2 that each start at the same time and have a different Pid than the parent data flow GroupBy_DF.
Scenario 2: Run multiple sub data flows with DOP greater than 1

When Degree Of Parallelism (DOP) is set to a value greater than 1, each transform defined in the data flow replicates for use on a parallel subset of data. For more information, see *Degree of parallelism* on page 87.

Set DOP to a value greater than 1 on the data flow Properties window.
The following diagram shows the sub data flows that Data Services generates for GroupBy_DOP2_Job when the Run GROUP BY as a separate process is selected and DOP set to 2.

When you execute the job, the Trace Log shows that the software creates sub data flows that execute in parallel with different process IDs (Pids). For example, the following trace log shows the following four sub data flows that
start concurrently and that each have a different Pid than the parent data flow GroupBy_DOP2_DF:

- GroupBy_DOP2_DF_1_1
- GroupBy_DOP2_DF_1_2
- GroupBy_DOP2_DF_1_3
- GroupBy_DOP2_DF_1_4
Tip:
When your data flow has DOP is greater than one, select either job or data flow for the Distribution level option when you execute the job. If you
execute the job with the value sub data flow for Distribution level, the Round-Robin Split or Hash Split sends data to the replicated queries that might be executing on different job servers. Because the data is sent on the network between different job servers, the entire data flow might be slower. For more information about job distribution levels, see Using grid computing to distribute data flow execution on page 118.

Data_Transfer transform

The Data_Transfer transform creates transfer tables in datastores to enable the software to push down operations to the database server. The Data_Transfer transform creates two sub data flows and uses the transfer table to distribute the data from one sub data flow to the other sub data flow. The sub data flows execute serially.

Related Topics
• Reference Guide: Transforms, Data_Transfer

Examples of multiple processes with Data_Transfer

The following are typical scenarios of when you might use the Data_Transfer transform to split a data flow into sub data flows to push down operations to the database server.

Scenario 1: Sub data flow to push down join of file and table sources

Your data flow might join an Orders flat file and a Orders table, perform a lookup_ext function to obtain sales subtotals, and another Query transform to group the results by country and region.
To define sub data flows to push down a join of a file and table

1. Add a Data_Transfer transform between the Orders file source and the Query transform.
2. Select the value **Table** from the drop-down list in the **Transfer type** option in the Data_Transfer editor.

![Schema: Orders_Flatfile](image)

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
<th>Business Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERID</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>ORDERDATE</td>
<td>varchar(11)</td>
<td></td>
</tr>
<tr>
<td>UNITPRICE</td>
<td>decimal(10,3)</td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>PRODUCTID</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>varchar(50)</td>
<td></td>
</tr>
<tr>
<td>COMPNAME</td>
<td>varchar(50)</td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td>varchar(25)</td>
<td></td>
</tr>
<tr>
<td>COUNTRY</td>
<td>varchar(25)</td>
<td></td>
</tr>
</tbody>
</table>

![General Options](image)

- **Enable transfer**: Yes
- **Transfer type**: Table
- **Join rank**: 0
- **File options**:
  - **Root directory**: ...
  - **File name**: ...
- **Table options**:
  - **Table name**: Northwind_DS.DBO.ORDERS_FROMFILE
  - **Database type**: Microsoft SQL Server 2000
  - **Array fetch size**: 1000

3. For **Table name** in the Table options area, browse to the datastore that contains the source table that the Query joins to this file. Double-click the datastore name and enter a name for the transfer table on the Input table for Data_Transfer window.

In this example, browse to the same datastore that contains the Orders table and enter Orders_FromFile in Table name.
4. After you save the data flow and click **Validation Display Optimized SQL...**, the Optimized SQL window shows that the join between the transfer table and source Orders table is pushed down to the database.

![Optimized SQL window](image)

SAP BusinessObjects Data Services can push down many operations without using the Data_Transfer transform. For more information, see *Push-down operations* on page 42.

5. When you execute the job, the Trace Log shows messages such as the following that indicate that the software created two sub data flows with...
different Pids to run the different operations serially.

Scenario 2: Sub data flow to push down memory-intensive operations

You can use the Data_Transfer transform to push down memory-intensive operations such as Group By or Order By.

For the sample data flow in Scenario 1: Sub data flow to push down join of file and table sources on page 112, you might want to push down the Group By operation.
To define sub data flows to push down another operation

1. Add a Data_Transfer transform between the Lookup and GroupBy query transforms, as the following diagram shows.

   ![Diagram showing data flow with Data_Transfer, Query, Lookup, GroupBy, and Data_Transfer]

2. Select the value Table from the drop-down list in the Transfer type option in the Data_Transfer editor.

3. For Table name in the Table options area, browse to the datastore that contains the target table. Double-click the datastore name and enter a name for the transfer table on the Input table for Data_Transfer window.

4. After you save the data flow and click ValidationDisplay Optimized SQL..., the Optimized SQL window shows that the software pushes the GroupBy down to the target database.

   ![Optimized SQL window showing SQL code]

   ```sql
   INSERT INTO "dbo"."JOINTARGET" ("PRODUCTID", "SHIPCOUNTRY", "SHIREGION", "SALES")
   SELECT "Data_Transfer_1.Lookup"."PRODUCTID", "Data_Transfer_1.Lookup"."SHIPCOUNTRY", "Data_Transfer_1.Lookup"."SHIREGION", sum("Data_Transfer_1.Lookup"."SALES")
   FROM "dbo"."GROUPTRANS" "Data_Transfer_1.Lookup"
   JOIN "dbo"."JOINTARGET" ON "Data_Transfer_1.Lookup"."PRODUCTID" = "JOINTARGET"."PRODUCTID" AND "Data_Transfer_1.Lookup"."SHIPCOUNTRY" = "JOINTARGET"."SHIPCOUNTRY" AND "Data_Transfer_1.Lookup"."SHIREGION" = "JOINTARGET"."SHIREGION"
   ```
The software can push down many operations without using the Data_Transfer transform. For more information, see *Push-down operations* on page 42.

5. When you execute the job, the messages indicate that the software creates three sub data flows to run the different operations serially.

### Using grid computing to distribute data flow execution

SAP BusinessObjects Data Services takes advantage of grid computing when you:

- Define a group of Job Servers (called a *Server Group* on page 118) that acts as a server grid. The software leverages available CPU and memory on the computers where the Job Servers execute.

- Specify *Distribution levels for data flow execution* on page 118 to process smaller data sets or fewer transforms on different Job Servers in a Server Group. Each data flow or sub data flow consumes less virtual memory.

#### Server Group

You can distribute the execution of a job or a part of a job across multiple Job Servers within a Server Group to better balance resource-intensive operations. A server group automatically measures resource availability on each Job Server in the group and distributes scheduled batch jobs to the Job Server with the lightest load at runtime.

**Related Topics**

- *Management Console Administrator Guide: Server Groups*

### Distribution levels for data flow execution

When you execute a job, you can specify the following values on the *Distribution level* option:

- Job level - An entire job can execute on an available Job Server.
• Data flow level - Each data flow within a job can execute on an available Job Server and can take advantage of additional memory (up to two gigabytes) for both in-memory and pageable cache on another computer.

• Sub data flow level - A resource-intensive operation (such as a sort, table comparison, or table lookup) within a data flow can execute on an available Job Server. Each operation can take advantage of up to two gigabytes additional memory for both in-memory and pageable cache on another computer.

Job level

When you choose a Server Group to execute your job, the default distribution level is Job, as the following Execution Properties window shows.
When **Distribution level** has the value **Job**, all of the processes that belong to the job execute on the same computer. For example, section **Scenario 2: Run multiple sub data flows with DOP greater than 1** on page 108 describes the data flow **GroupBy_DOP2_DF** which is designed to generate four sub data flows as follows:

Starting sub data flow **<GroupBy_DOP2_DF_1_1>** on job server host **<SJ-C>**, port **<3502>**. Distribution level **<Job>**.
Starting sub data flow **<GroupBy_DOP2_DF_1_2>** on job server host **<SJ-C>**, port **<3502>**. Distribution level **<Job>**.
Starting sub data flow **<GroupBy_DOP2_DF_1_3>** on job server host **<SJ-C>**, port **<3502>**. Distribution level **<Job>**.
Starting sub data flow **<GroupBy_DOP2_DF_1_4>** on job server host **<SJ-C>**, port **<3502>**. Distribution level **<Job>**.

When **Distribution level** is **Job**, the software uses named pipes to send data between the sub data flow processes on the same computer, as the following diagram indicates with the blue arrows.
Data flow level

When **Distribution level** has the value **Data flow**, all of the processes that belong to each data flow can execute on a different computer. For example, the following `GroupBy_Q1_Q2_Job` has two data flows: `GroupQ1_DF` and `GroupQ1_DF` that process orders for the first quarter and second quarter, respectively.

- The solid blue lines enclose each process that can execute on a separate Job Server. In this example, each data flow can execute on a different computer than the computer where the job started.

- SAP BusinessObjects Data Services uses Inter-Process Communications (IPC) to send data between the job and data flows on the different computers, as the diagram indicates with the yellow arrows. IPC uses the peer-to-peer port numbers specified on the **Start port** and **End port** options in the Server Manager.

**Note:**
The default values for **Start port** and **End port** are 1025 and 32767, respectively. Change these values if you want to restrict the number of ports or if some of the ports are already in use.

When you execute the job, the Trace log displays messages such as the following that indicate the communication port for the data flow and the
distribution level for each data flow. All of the sub data flows within a data flow run on the same computer.

Data flow communication using peer-to-peer method with the port range <1025> to <32767>.

... Peer-to-peer connection server for session process is listening at host <SJ-C>, port <1025>. Job <GroupBy_Q1_Q2_Job> is started. Starting data flow </GroupBy_Q1_Q2_Job/GroupBy_Q1_DF> on job server host <SJ-C>, port <3502>. Distribution level <Data flow>. Data flow submitted to server group <sg_direpo>. Load balancing algorithm <Least load>. Server group load statistics from job server <mssql_lap_js SJ-C 3502>:  
<sql_lap_js SJ-C 3502> System Load <47%> Number of CPUs <1>  
<MSSQL2005_JS SJ-W-C 3500> System Load <70%> Number of CPUs <2>  
Process to execute data flow <GroupBy_Q1_DF> is started. Starting sub data flow <GroupBy_Q1_DF_1_1> on job server host <SJ-C>, port <3502>. Distribution level <Data flow>. Starting sub data flow <GroupBy_Q1_DF_1_2> on job server host <SJ-C>, port <3502>. Distribution level <Data flow>. Starting sub data flow <GroupBy_Q1_DF_1_3> on job server host <SJ-C>, port <3502>. Distribution level <Data flow>. Starting sub data flow <GroupBy_Q1_DF_1_4> on job server host <SJ-C>, port <3502>. Distribution level <Data flow>.

Sub data flow level

When Distribution level has the value Sub data flow, each sub data flow within a data flow can execute on a different computer. In the example that section Scenario 2: Run multiple sub data flows with DOP greater than 1 on page 108 describes, the GroupBy_DOP2_Job has four sub data flows as follows.
• The solid blue lines enclose each process that can execute on a separate Job Server. In this example, each sub data flow can execute on a different computer than the computer where the job started.

• The yellow arrows indicate the Inter-Process Communications (IPC) that SAP BusinessObjects Data Services uses to send data between the job and sub data flows on the different computers. IPC the peer-to-peer port numbers specified on the Start port and End port options in the Server Manager.

The default values for Start port and End port are 1025 and 32767, respectively. Change these values if you want to restrict the number of ports or if some of the ports are already in use.

**Note:**
If you find that sending data across the network is causing your data flow to execute longer, you might want to change Distribution level from Sub data flow to Data flow or Job.

When you execute the job, the Trace log displays messages such as the following that indicate that the software selects a job server for each sub data flow based on the system load on each computer:

Starting sub data flow <GroupBy_DOP2_DF_1_1> on job server host <SJ-C>, port <3502>. Distribution level <Sub data flow>. Sub data flow submitted to server group <sg_direpo>. Load balancing algorithm <Least load>. Server group load statistics from job server <mssql_lap_js SJ-C 3502>:
<mssql_lap_js SJ-C 3502> System Load <21%> Number of CPUs <1>
<MSSQL2005_JS SJ-W-C 3500> System Load <70> Number of CPUs <1>
Starting sub data flow <GroupBy_DOP2_DF_1_2> on job server host <SJ-C>, port <3502>. Distribution level <Sub data flow>. Sub data flow submitted to server group <sg_direpo>. Load balancing algorithm <Least load>. Server group load statistics from job server <mssql_lap_js SJ-C 3502>:
<mssql_lap_js SJ-C 3502> System Load <21%> Number of CPUs <1>
<MSSQL2005_JS SJ-W-C 3500> System Load <70> Number of CPUs <2>

The following messages show the communication port that each sub data flow uses:

Peer-to-peer connection server for sub data flow <Group By_DOP2_DF_1_1> is listening at host <SJ-C>, port <1027>. Process to execute sub data flow <GroupBy_DOP2_DF_1_4> is started.
Peer-to-peer connection server for sub data flow <Group
By_DOP2_DF_1_2> is listening at host <SJ-C>, port <1028>. Peer-to-peer connection server for sub data flow <Group By_DOP2_DF_1_3> is listening at host <SJ-C>, port <1029>. Peer-to-peer connection server for sub data flow <Group By_DOP2_DF_1_4> is listening at host <SJ-C>, port <1030>.
Using Bulk Loading
SAP BusinessObjects Data Services supports capabilities present in Oracle, Microsoft SQL Server, HP Neoview, Informix, DB2, Netezza, Sybase ASE, Sybase IQ, and Teradata that enable you to load data in bulk rather than using SQL statements. Some general considerations when using bulk loading are:

- You specify bulk-loading options on the Data Services target table editor in the **Bulk Loader Options** tab.
- If you are using bulk loading, parallel loading is available only in the Oracle and Informix environment.
- When you use a template table, bulk loading is disabled.

**Related Topics**
- *Reference Guide: Data Services Objects, Target*

---

**Bulk loading in Oracle**

SAP BusinessObjects Data Services supports Oracle bulk loading.

**Bulk-loading methods**

You can bulk load to Oracle using an API or a staging file:

- If you select the **API** method, SAP BusinessObjects Data Services accesses the direct path engine of Oracle's database server associated with the target table and connected to the target database. Using Oracle's
Direct-Path Load API, input data feeds directly into database files. To use this option, you must have Oracle version 8.1 or later.

- If you select the **File** method, Data Services writes an intermediate staging file, control file, and log files to the local disk and invokes the Oracle SQL*Loader. This method requires more processing time than the API method.

For detailed information about the Oracle SQL*Loader options, see the relevant Oracle product documentation.

**Bulk-loading modes**

Bulk loading in Oracle supports two modes of data loading: conventional-path and direct-path. Conventional-path loading is implicit for the **File** option if you do not select **Direct-path** on the **Bulk Loader Options** tab in the target table editor. SAP BusinessObjects Data Services always uses direct-path loading for the **API** option.

- **Conventional-path loading**

  Conventional-path loads use the SQL INSERT statements to load data to tables.

- **Direct-path loading**

  Direct-path loads use multiple buffers for a number of formatted blocks that load data directly to database files associated with tables.

**Bulk-loading parallel-execution options**

Parallel-execution options for bulk loading are on the **Options** tab.

For the API method, you can choose to select the **Enable partitioning** check box. If selected, SAP BusinessObjects Data Services generates the number of target parallel instances based on the number of partitions in the target table. If not selected or if your table target is not partitioned, Data Services uses one loader by default.

For the File method, enter a value in the **Number of loaders** box or select the **Enable partitioning** check box.
**Note:**
The Enable partitioning check box does not appear on the Options tab if the target table is not partitioned.

### Bulk-loading scenarios

With two bulk-loading methods, two load modes, and two parallel load options, there are several scenarios you can configure:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Method</th>
<th>Load Mode</th>
<th>Parallel Load Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>API</td>
<td>Direct-path</td>
<td>Enable partitions is not selected (One loader is used by default)</td>
</tr>
<tr>
<td>2</td>
<td>API</td>
<td>Direct-path</td>
<td>Enable partitions is selected</td>
</tr>
<tr>
<td>3</td>
<td>File</td>
<td>Direct-path</td>
<td>Number of loaders = 1</td>
</tr>
<tr>
<td>4</td>
<td>File</td>
<td>Direct-path</td>
<td>Number of loaders &gt; 1</td>
</tr>
<tr>
<td>5</td>
<td>File</td>
<td>Direct-path</td>
<td>Enable partitions is selected</td>
</tr>
<tr>
<td>6</td>
<td>File</td>
<td>Conventional</td>
<td>Number of loaders = 1</td>
</tr>
<tr>
<td>7</td>
<td>File</td>
<td>Conventional</td>
<td>Number of loaders &gt; 1</td>
</tr>
</tbody>
</table>
Here are some tips for using these scenarios:

• The API method always uses the direct-path load type, and when it is used with a partitioned target table, SAP BusinessObjects Data Services processes loads in parallel. The software instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition.

• With the File method, direct-path is faster than conventional load, but the File method is slower than using an API because of the need to generate a staging file, logs, and invoke Oracle's SQL*Loader.

• With the File method, when you use a value of greater than one for either the Number of Loaders or the Enable partitioning option, loads cannot truly run in parallel. The creation of a staging file and log for each loader is serialized.

Using bulk-loading options

As seen in the table on the previous page, there are many ways to set up bulk loading for an Oracle database. The following sections describe two scenarios in detail.

Direct-path loads using Number of Loaders and File method

In the Options tab of the target table editor, when you enter a value for Number of loaders, SAP BusinessObjects Data Services instantiates multiple loaders. Each loader receives rows equal to the amount specified in the Rows per commit box on the Bulk Loader Options tab. The loaders pipe rows to a staging file, then call the SQL*Loader to load the staging file contents into the table.
This process occurs in "round-robin" fashion. For example, if you set **Rows per commit** to 5000 and **Number of loaders** to 2, then the first loader receives 5000 rows, writes them to the staging file, and then invokes the SQL*Loader to load the data into the table.

Meanwhile, the second loader receives the second batch of 5000 rows, writes them to a staging file, and then waits for the first loader to complete the loading. When the first loader completes the bulk load, the second loader starts, and while the second loader is loading, the first loader receives the third batch of 5000 rows. This process continues until all the data loads.

The SQL*Loader uses a control file to read staging files and load data. The software either creates this control file at runtime or uses one that is specified on the **Bulk Loader Options** tab at design time.

For parallel loading, the generated control files, data files, and log files are named as follows:

```
TableNameTIDPID_LDNUM_BATCHNUM
```

Where:

* **TableName**: The name of the table into which data loads.

* **TID**: The thread ID.

* **PID**: The process ID.

* **LDNUM**: The loader number, which ranges from 0 to number of loaders minus 1. For single loaders, **LDNUM** is always 0.

* **BATCHNUM**: The batch number the loader is processing. For single loaders the **BATCHNUM** is always 0.

**Note:**
Product performance during this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.
Direct-path loads using partitioned tables and API method

You can import partitioned tables as SAP BusinessObjects Data Services metadata.

In the **Options** tab of the target table editor, when you select **Enable partitioning**, the software instantiates multiple loaders based on the number of partitions in a table. Each loader receives rows that meet the conditions specified by the partition. In addition, commits occur based on the number specified in the **Rows per commit** box on the **Bulk Loader Options** tab.

For example:

- If you **Rows per commit** to 5000, the number of partitions is set to 2, and your first partition includes 2500 rows, then the first loader commits after receiving all possible rows (2500) while concurrently processing the second loader.

- If you **Rows per commit** to 5000, the number of partitions is set to 2, and your first partition includes 10,000 rows, then the first loader commits twice. Once after receiving 5000 rows and again after receiving the second batch of 5000 rows. Meanwhile, the second loader is processing its rows.

The loaders pipe rows directly to Oracle database files by using Oracle direct-path load APIs (installed with the Oracle client) that are associated with the target database.

The API method allows the software to bypass the use of the SQL* Loader (and the control and staging files it needs). In addition, by using table partitioning, bulk loaders can pass data to different partitions in the same target table at the same time. Using the API method with partitioned tables fully optimizes performance.

**Note:**
If you plan to use a partitioned table as a target, the physical table partitions in the database must match the metadata table partitions in SAP BusinessObjects Data Services. If there is a mismatch, Data Services will not use the partition name to load partitions, which impacts processing time.

For the API method, the software records and displays error and trace logs as it does for any job. A monitor log records connection activity between components; however, it does not record activity while the API is handling the data.
Bulk loading in Microsoft SQL Server

SAP BusinessObjects Data Services supports Microsoft SQL Server bulk loading through the SQL Server ODBC bulk copy API. For detailed information about the SQL Server ODBC bulk copy API options and their behavior in the Microsoft SQL Server DBMS environment, see the relevant Microsoft SQL Server product documentation.

To use the SQL Server ODBC bulk copy API

1. From the Tools menu, select Options > Job Server > General.
2. For Section, enter al_engine.
3. For Key, enter UseSQLServerBulkCopy.
4. Select TRUE (default) or FALSE. If you leave the default, the software uses the SQL Server ODBC bulk copy API. If you set this parameter to FALSE, the software overrides the default and uses the SQLBulkOperations API.

Network packet size option

When loading to SQL Server, the client caches rows until it either fills a network packet or reaches the commit size (regardless of whether the packet is full). Then the client sends the packet to the server. You can affect performance by tuning commit size and network packet size. You can change these sizes on the Bulk Loader Options tab for SQL Server:

- Rows per commit

  This option lets you specify the number of rows to put in the cache before issuing a commit.

- Network packet size

  This option lets you specify network packet size in kilobytes. The default packet size is 4 kilobytes.
Note:
It is recommended that you set the **Rows per commit** and **Network packet size** parameters to avoid sending many partially filled packets over the network and ensure that the packet size contains all rows in the commit.

### Maximum rejects option

The **Maximum rejects** parameter (on the Bulk Loader Options page) can also affect your SQL Server bulk-loading performance. When you set **Maximum rejects** to 0, SAP BusinessObjects Data Services stops at the first error it encounters and does not cache rows in the transaction (caching rows in a transaction allows the software to process each row even if an error occurs during the transaction commit process.)

When you do not specify a value for **Maximum rejects**, the software ignores the rejected rows, logs warnings, and continues processing.

### Bulk loading in HP Neoview

SAP BusinessObjects Data Services supports bulk loading to HP Neoview via Neoview Transporter.

For detailed information about HP Neoview loading options and their behavior, see the relevant HP Neoview product documentation.

To use Neoview Transporter, you must also install the following components:

- Neoview Transporter Java Client
- Java JRE version 1.5 or newer
- Neoview JDBC Type 4 Driver
- Neoview ODBC Windows Driver (for Windows)
- Neoview UNIX Drivers (for connecting to a database on UNIX)
- Neoview Command Interface

Note:

- If you are using multibyte data on Windows, you must change the Windows regional settings to the multibyte language, for example, Japanese.
- When you install the Java Client, an environment variable called NVTHOME is created and will point to the location of the Neoview
Transporter base directory. You may receive an error in SAP BusinessObjects Data Services if NVTHOME is not defined.

HP Neoview recommends that you use the bulk-loading method to load data for faster performance. The SAP BusinessObjects Data Services bulk loader for HP Neoview supports UPDATE and UPSERT as well as INSERT operations, which allows for more flexibility and performance.

SAP BusinessObjects Data Services generates a control file as input into Neoview Transporter. The control file specifies the data files and the target tables to be loaded. Being in UTF-8, the control file supports multibyte data.

By default, HP Neoview uses the SQL insert operation. For SQL update and upsert options, the control file specifies the columns used in the WHERE clause and the columns to be updated in the UPDATE clause. By default, SAP BusinessObjects Data Services uses the primary key columns in the WHERE clause.

To bulk load to a HP Neoview target table, the software:

• Creates a control file in UTF-8
• Loads data from the source into the file or named pipe in UTF-8
• Invokes Neoview Transporter

How Data Services and HP Neoview use the file options to load

You can choose to use either named pipes or data files as staging for loading the data. Choose from the following file options:

• Data file (for Windows and UNIX)
• Named pipe (for UNIX only)

Data file

SAP BusinessObjects Data Services runs bulk-loading jobs using a staging data file as follows:

1. The software generates staging data file(s) containing data to be loaded into a HP Neoview table.
2. The software generates a control file to be used by Neoview Transporter.

**Named pipe**

SAP BusinessObjects Data Services runs bulk-loading jobs using named pipes as follows:

1. The software generates a control file that Neoview Transporter uses to manipulate the database.
2. The software creates a pipe to contain the data to apply into an HP Neoview table.

On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format:

```plaintext
/temp/filename.dat
```

3. The software invokes Neoview Transporter with the control file as input.
4. The software writes data to the pipes.
5. Neoview Transporter reads data from the pipe and applies data to the HP Neoview table.

**Using the UPSERT bulk operation**

The purpose of the HP Neoview Upsert operation is to update a row, but if no row matches the update, the row is inserted.

In SAP BusinessObjects Data Services, you enable Upsert on the target table editor's **Bulk Loader Options** tab. In the **Data Services options** section, for **Bulk Operation**, select **Upsert** (the default is **Insert**) in the SQL Operation list.

After selecting **Upsert**, notice you can also enable the **Use input keys** option on the target editor's **Options** tab. The **Use input keys option** will assign the input primary keys as primary keys in the target table.

**Related Topics**

- Reference Guide: Objects, Target tables
Bulk loading in Informix

SAP BusinessObjects Data Services supports Informix bulk loading. For detailed information about Informix bulk-loading utility options and their behavior in the Informix DBMS environment, see the relevant Informix product documentation.

Setting up Informix for bulk-loading requires that you set the environment variable `INFORMIXDIR` at the system level even though the job service is configured to log on through an account with that variable defined at a user level.

To set Informix server variables

1. Right-click My Computer and choose Properties.
2. From the Properties dialog, click the Environment tab.
3. From the Environment tab, add the following to both the System variable and User variable (where `c:\Informix` is the installation directory):

   ```
   INFORMIXDIR=c:\Informix
   INFORMIXSERVER=c:\Informix
   ```

SAP BusinessObjects Data Services provides Informix bulk-loading support only for single-byte character ASCII delimited files (not for fixed-width files).

For SAP BusinessObjects Data Services to initiate Informix bulk loading directly, the Job Server and the target database must be located on the same system.

Bulk loading in DB2 Universal Database

SAP BusinessObjects Data Services supports bulk loading to the DB2 Universal Database in the following ways:

- Using the DB2 CLI load method on page 138
- Using the DB2 bulk load utility on page 139
- Using the import utility on page 144
When to use each DB2 bulk-loading method

SAP BusinessObjects Data Services supports multiple bulk-loading methods for DB2 Universal Database (UDB) on Windows and UNIX. The following table lists the methods that you can select depending on your requirements.

Note:
You cannot bulk load data to DB2 databases that run on AS/400 or z/OS (MVS) systems.

<table>
<thead>
<tr>
<th>Load method</th>
<th>Description</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| CLI Load    | Loads a large volume of data at high speed by passing it directly from memory to the table on the DB2 UDB server. | • Provides the fastest way to bulk load.  
• Eliminates some parameters because no intermediate data file is required.  
• Can put rows that violate the unique key constraint into an exception table. | • Must specify Recoverable and Copy target directory options to enable recovery because DB2 logging is not enabled for CLI Load.  
• The DB2 UDB server and client must be Version 8.0 or later.  
• Stops loading when it encounters the first rejected row. |
| Load        | Loads a large volume of data by writing to a data file that it passes to the DB2 UDB server. | • This method is faster than the import utility.  
• Puts rejected rows into a "dump" file.  
• Can put rows that violate unique key constraint into an exception table. | • Must have disk space for intermediate data file.  
• Must specify Recoverable and Copy target directory options to enable recovery because DB2 logging is not enabled for DB2 Load.  
• The DB2 UDB server and client must be Version 8.0 or later. |
<table>
<thead>
<tr>
<th>Load method</th>
<th>Description</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>Loads a large volume of data by using a SQL INSERT statement to write data from an input file into a table or view.</td>
<td>• Recovery is enabled automatically because DB2 logging occurs during import. • Performs referential integrity or table constraint checking in addition to unique key constraint checking.</td>
<td>• Because DB2 logs each INSERT statement, this method is the slowest way to bulk load data. • The Data Services Job Server and DB2 UDB server must be on the same computer.</td>
</tr>
</tbody>
</table>

**Using the DB2 CLI load method**

The DB2 Call Level Interface (CLI) load method performs faster than the bulk load or import utilities because it does not write the data to an intermediate file. Instead, the CLI load method writes the data from memory (where SAP BusinessObjects Data Services extracted or transformed the data) directly to the table on the DB2 server.

**To configure your system to use the CLI load method**

1. Enter the appropriate information in the datastore editor, on the **DB2 Properties** tab.
   Fields include:
   • **Bulk loader user name**—The user name SAP BusinessObjects Data Services uses when loading data with the CLI load option. For bulk loading, you might specify a different user name, for example one with import and load permissions.
   • **Bulk loader password**—The password SAP BusinessObjects Data Services uses when loading with the CLI load option.
2. To use a different bulk loader working directory than the default ($LINK_DIR\log\bulkloader), specify the directory name in the datastore editor on the *Connections* tab.

**To use the CLI load method in a job**

1. Open the DB2 target table editor in the Designer workspace.
2. Select the *Bulk Loader Options* tab below the table schema area.
3. In the *Bulk loader* list, select **CLI load**.

   The window updates to show all CLI load options. CLI load options include these existing bulk load options:
   
   • Mode
   • Warning row count
   • Exception table name
   • Recoverable
   • Copy target directory

Additional or changed CLI load options include:

• **Maximum bind array** — Defines the maximum number of rows extracted or transformed before the software sends the data to the DB2 table or view. If you do not enter a value, Data Services uses the CLI load default value of 10000 rows.

• **Clean up bulk loader directory** — If you select this option, the software deletes the message file when the CLI load completes successfully. Because the CLI load obtains the data from memory, Data Services creates no control or data files.

**Related Topics**

• *Reference Guide: Objects, Target tables*

**Using the DB2 bulk load utility**

The DB2 load utility performs faster than the import utility because it writes data directly into the data file.
To configure your system to use the load utility

1. Connect to the DB2 Version 8.x target database that uses the following:
   - For HP-UX11i only, run:
     
     ```
     bind <LINK_DIR>/bin/db2bulkload8.bnd blocking all grant public
     ```
   - For all other platforms, run:
     
     ```
     bind <LINK_DIR>/bin/db2bulkload.bnd blocking all grant public
     ```

2. Determine how SAP BusinessObjects Data Services will transmit data files to DB2. Depending on your configuration, there are different ways to transmit data files.

   For example, if your Data Services Job Server and the DB2 server reside on different computers, you can choose one of the following methods:
   - FTP— SAP BusinessObjects Data Services generates the data file and uses FTP to send the file to the DB2 server. To use the FTP option, you must define the FTP host name, user name, and password in the DB2 datastore you create in the Designer.
   - Data file on DB2 client computer— SAP BusinessObjects Data Services writes to the data file on the DB2 client computer, and the DB2 client transfers the data directly to the DB2 server during the load process. To use this option, you must select Data file on client computer on the Bulk Loader Options tab when you define the target in your job.

The following matrix outlines supported data file transmission methods.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Data file transmission methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects Data Services and DB2 server on same computer</td>
<td>Automatic data transmission</td>
</tr>
</tbody>
</table>
### Configuration

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Data file transmission methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP BusinessObjects Data Services and DB2 server on different computer</td>
<td>FTP or data file on DB2 client computer.</td>
</tr>
</tbody>
</table>

3. If SAP BusinessObjects Data Services and DB2 are on different computers, you must provide a working directory for SAP BusinessObjects Data Services on the DB2 server. SAP BusinessObjects Data Services instructs the DB2 load process to generate the file for rejected rows on the DB2 server at this location.

4. Enter the appropriate information in the datastore editor, on the **DB2 Properties** tab. Fields include:
   - **Bulk loader user name**—The user name the software uses when loading data with the bulk loader option. For bulk loading, you might specify a different user name—one who has import and load permissions, for example.
   - **Bulk loader password**—The password the software uses when loading with the bulk loader option.
   - **FTP host name**—If this field is left blank or contains the name of the computer where the Job Server resides, the software assumes that DB2 and SAP BusinessObjects Data Services share the same computer and that FTP is unnecessary. When FTP is unnecessary, all other FTP-related fields can remain blank.
   - **FTP user name**—Must be defined to use FTP.
   - **FTP password**—Must be defined to use FTP.
   - **Server working directory**—The working directory for the load utility on the computer that runs the DB2 server. You must complete this field whenever the DB2 server and the Job Server run on separate computers. SAP BusinessObjects Data Services instructs the DB2 load process to generate the file for rejected rows on the DB2 server at this location.

5. If SAP BusinessObjects Data Services will use FTP and the DB2 server runs on Windows NT, verify connectivity with the FTP server.
If your Job Server runs on Windows NT:

- Connect to the FTP server using the command:

```bash
ftp <ServerName>
```

- Type the command:

```bash
put <LINK_DIR>\DSConfig.txt <Server working directory>\DSConfig.txt
```

where `<LINK_DIR>` is the SAP BusinessObjects Data Services installation directory and `<Server working directory>` is the working directory entered on the datastore’s DB2 Properties tab.

You can only use the load utility if this command succeeds.

If your Job Server runs on UNIX:

- Connect to the FTP server.

- Change directories to the `Server working directory` entered on the `DB2 Properties` tab on the datastore editor.

For example, if the directory is `c:\temp`, type:

```bash
cd c:\temp
```

You can only use the load utility if this command succeeds.

6. To use a different bulk loader working directory than the default (`$LINK_DIR\log\bulkloader`), specify the directory name in the datastore editor on the Connections tab.

**To use the load utility in a job**

1. Open the DB2 target table editor in the Designer workspace.
2. Select the **Bulk Loader Options** tab below the table schema area.
3. In the **Bulk loader** list, select **load**.

The window updates to show all load options. Load options include these existing import bulk loader options:

- **Generate files only**
- **Clean up bulk loader directory**
• Text delimiter
• Column delimiter

Additional load options include:
• **Mode**—Determines load mode. Valid values are:
  • **Insert**—Appends the new records into the target table
  • **Replace**—Deletes the existing records, then inserts the loaded data
• **Save count**—Determines the consistency point while loading data into tables.
• **Warning row count**—Defines the number of warnings allowed for each load operation.
• **Exception table name**—Defines the table into which the DB2 server loads rows that violate a table defined with constraints. Rows that violate those constraints are deleted from the target table and inserted into the exception table.
• **Recoverable**—Enables or disables data recovery.
  • When this option is not selected (disabled), the load operation is not recoverable because DB2 does not log the loading of data.
  • When selected (enabled), DB2 makes a copy of the loaded portion of the table. DB2 uses this copy in the roll-forward phase of database recovery. You must define the path in the **Copy target directory** for this copy.
• **Copy target directory**—Defines the directory of the copy files when you enable both the database forward log recovery and select the **Recoverable** option. SAP BusinessObjects Data Services supports only the copy files option for the DB2 CLI load method.
• **Data file on client computer**—When you select this option, the software writes to the data file on the DB2 client computer. SAP BusinessObjects Data Services does not need to FTP the data file because the DB2 client transfers the data directly to the DB2 server during the load process. To use this option:
  • You must use DB2 Version 8.x or later.
  • The target DB2 cannot be a DB2 enterprise (extended edition environment).
• The target table and database must not be partitioned.
• This option is only applicable if SAP BusinessObjects Data Services and DB2 are on different servers.

When you execute the DB2 bulk load utility, DB2 automatically generates the following files:
• Local message file (named .log) in the bulk loader working directory. DB2 writes output messages into this log file.
• "Dump" file (named .bad) in the DB2 server working directory. DB2 writes rejected input rows into this .bad file. If you clear the Data file on client machine option, SAP BusinessObjects Data Services uses FTP to send the .bad file to the bulk loader working directory and deletes it after the load completes successfully.

Check the trace log to find either of these files.

Related Topics
• Reference Guide: Objects, Target tables (Teradata target table options)

Using the import utility

SAP BusinessObjects Data Services also supports bulk loading in the DB2 Universal Database 5.2 environment using the import utility. For the software to initiate DB2 bulk loading by this method directly, the Job Server and DB2 must be located on the same system. If they are not, use the following procedure to initiate bulk loading:

1. Generate a control file and data file. Check Generate files only in the target table editor on the Bulk Loader Options tab.
2. Manually move the control file and data file to the system where the target database is located.
3. Start the execution of the bulk loader manually.

Bulk loading in Netezza

SAP BusinessObjects Data Services supports bulk loading to Netezza Performance Servers.
For detailed information about Netezza loading options and their behavior in the Netezza environment, see the relevant Netezza product documentation.

Netezza recommends using the bulk-loading method to load data for faster performance. Unlike some other bulk loaders, the SAP BusinessObjects Data Services bulk loader for Netezza supports UPDATE and DELETE as well as INSERT operations, which allows for more flexibility and performance.

### Netezza bulk-loading process

To bulk load to a Netezza target table, SAP BusinessObjects Data Services:

- Creates an external table that is associated with a local file or named pipe
- Loads data from the source into the file or named pipe
- Loads data from the external table into a staging table by executing an INSERT statement
- Loads data from the staging table to the target table by executing a set of INSERT/UPDATE/DELETE statements

#### Options overview

From the **Bulk Loader Options** tab of your Netezza target table editor, select one of these methods depending on your Netezza environment:

- **Named pipe**— SAP BusinessObjects Data Services streams data as it is written to the named pipe through the external table to the staging table. For files that are larger than 4 GB in size, select this option for faster performance.
• **File**— SAP BusinessObjects Data Services writes the data to a file before loading through the external table to the staging table. For files that are smaller than 4 GB in size, select this option for faster performance.

• **None**— SAP BusinessObjects Data Services does not use bulk loading.

Because the bulk loader for Netezza also supports UPDATE and DELETE operations, the following options (on the target table editor **Options** tab) are also available for Netezza bulk loading.

**Note:**
These are the only options that will be available on the **Options** tab.

• Column comparison
• Number of loaders
• Use input keys
• Update key columns
• Auto correct load

**Related Topics**
• *Reference Guide: Objects, Target tables*

**Netezza log files**

When writing from the external table to the staging table, Netezza generates logs (nzlog and nzbad files) and writes them to a database server working directory. You can use these logs to troubleshoot your jobs. (If you do not enter a **Database server working directory** in the datastore editor, Netezza uses the temp directory on its server, /tmp, to store the nzlog and nzbad files.)

For SAP BusinessObjects Data Services to access and manage these logs, configure the FTP parameters in the datastore editor. After a load, the software copies these files from the specified Netezza **Database server working directory** to the specified **Bulk loader directory** and deletes them from the Netezza server.

For successful loads, SAP BusinessObjects Data Services then deletes these log files from the **Bulk loader directory** (assuming the **Clean up bulk loader directory after load** option is checked in the target table editor).
For failed loads, the software does not delete the log files from the Bulk loader directory even if the Clean up bulk loader directory after load option is checked in the target table editor.

**Configuring bulk loading for Netezza**

To configure the datastore:

1. Create a new ODBC datastore.
2. Specify the options for the Netezza database by selecting the Database type: ODBC.
3. Click Advanced.
4. Click in the field to the right of Bulk loader directory and type the directory path or click Browse to where the software should write SQL and data files for bulk loading.
5. Expand the FTP category and enter the FTP host name, login user name, login password, and host working directory.
   
   These options are used to transfer the Netezza nzlog and nzbad files.

   **Note:**
   If this datastore is not being used specifically for Netezza bulk loading, the software ignores any FTP option entries.

6. If you are loading non-ASCII character data, set the Code page to latin-9.
   If you are loading multibyte data, set the Code page to utf-8.

7. Click OK or Apply.

**Related Topics**
- Reference Guide: Objects, Database datastores (ODBC)
- Designer Guide: Datastores, Defining a database datastore

**To configure the target table**

For descriptions of all Netezza bulk-loading options.

1. Create a data flow with a Netezza target table.
2. Double-click the target table to open its editor.
3. Click the **Bulk Loader Options** tab.

4. For **Bulk load**, from the drop-down list select a bulk-loading method (**File** or **Named pipe**).

5. Edit the other options as necessary.

   Click the Options tab. For Netezza databases, only the following options are available. Edit them as necessary.

   - Column comparison
   - Number of loaders
   - Use input keys
   - Update key columns
   - Auto correct load

**Related Topics**

- *Reference Guide: Objects, Target tables*

---

### Bulk loading in Sybase ASE

SAP BusinessObjects Data Services supports bulk loading of Sybase ASE databases through the Sybase ASE bulk copy utility. For detailed information about the Sybase ASE bulk loader options and their behavior in the Sybase ASE DBMS environment, see the relevant Sybase ASE product documentation.

### Bulk loading in Sybase IQ

SAP BusinessObjects Data Services supports bulk loading Sybase IQ databases via the Sybase IQ LOAD TABLE SQL command. For detailed information about the Sybase IQ LOAD TABLE parameters and their behavior in the Sybase IQ database environment, see the relevant Sybase IQ product documentation.

If the job server and Sybase IQ database server are on the same computer, you can use named pipes for bulk loading instead of files. Using named pipes eliminates disk access time and usually improves the performance of the bulk loader.
If the job server and Sybase IQ database server are not on the same computer, Data Services generates a data file and transfers it via FTP to the database computer for loading. You set FTP parameters in the Sybase IQ datastore editor.

After a job executes, SAP BusinessObjects Data Services stores the Sybase IQ message and row logs in the **Bulk loader directory** specified in the datastore editor (regardless of the setting for the **JS and DB on same machine** option). A data file will also be present if you do not use the named pipe option.

- message log—Records constraint violations specified in the **Error handling** section of the target table **Bulk Loader Options** tab.

- row log—Contains the data from the violating row. The data in the row log is delimited by the **Field delimiter** character specified on the **Bulk Loader Options** tab.

If you select **Clean up bulk loader directory after load**, the software deletes the data file and log files after loading completes. If you choose not to clean up the bulk loader directory or if your job results in errors captured in the logs, the software does not delete the data file and log files.

### Bulk loading in Teradata

SAP BusinessObjects Data Services supports bulk loading with the Teradata Parallel Transporter application as well as with Teradata load utilities.

For detailed information about Teradata bulk loader options and their behavior in the Teradata DBMS environment, see the relevant Teradata product documentation.

SAP BusinessObjects Data Services supports multiple bulk-loading methods for the Teradata database. From the **Bulk Loader Options** tab of your Teradata target table editor, select one of these methods depending on your Teradata environment:

- Parallel Transporter method
- Teradata standalone utilities
  - MultiLoad
  - FastLoad
  - TPump
When to use each Teradata bulk load method

SAP BusinessObjects Data Services supports multiple bulk-loading methods for Teradata on Windows and UNIX. The following table lists the methods and file options that you can select, depending on your requirements.

<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Transporter</td>
<td>File — Loads a large volume of data by writing to a data file that it passes to the Teradata server.</td>
<td>• Can use SAP BusinessObjects Data Services parallel processing. • SAP BusinessObjects Data Services creates the loading script.</td>
<td>• The Teradata Server, Tools and Utilities must be Version 7.0 or later. • If you use TTU 7.0 or 7.1, see the Release Notes.</td>
</tr>
</tbody>
</table>
## Using Bulk Loading

### Bulk loading in Teradata

<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| Parallel Transporter | **Generic named pipe** — Loads a large volume of data by writing to a pipe from which Teradata reads. | • Provides a fast way to bulk load because:  
  • As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
  • Can use SAP BusinessObjects Data Services parallel processing.  
  • On Windows, no I/O to an intermediate data file occurs because a pipe is in memory  
  • SAP BusinessObjects-Data Services creates the loading script. | • A job that uses a generic pipe is not restartable.  
  • The Teradata Server, Tools and Utilities must be Version 7.0 or later.  
  • If you use TTU 7.0 or 7.1, see the Release Notes. |
<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| Parallel Transporter | **Named pipe access module** — Loads a large volume of data by writing to a pipe from which Teradata reads. | • The job is restartable. For details, see "Automatically recovering jobs" in the Designer Guide.  
• Provides a fast way to bulk load because:  
  • As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
  • Can use SAP BusinessObjects Data Services parallel processing.  
  • On Windows, no I/O to an intermediate data file occurs because a pipe is in memory  
  • SAP BusinessObjects Data Services creates the loading script. | • The Teradata Server, Tools and Utilities must be Version 7.0 or later.  
• If you use TTU 7.0 or 7.1, see the Release Notes. |
| Load utility | **File** — Loads data by writing to a data file that it passes to the Teradata server. | Load utilities are faster than INSERT statements through the ODBC driver. | • User must provide the loading script.  
• Cannot use SAP BusinessObjects Data Services parallel processing |
<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| Load utility       | Generic named pipe — Loads a large volume of data by writing to a pipe from which Teradata reads. | • Load utilities are faster than INSERT statements through the ODBC driver.  
• Named pipes are faster than data files because:  
  • As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
  • On Windows, no I/O to an intermediate data file occurs because a pipe is in memory | • User must provide the loading script.  
• Cannot use SAP BusinessObjects Data Services parallel processing  
• A job that uses a generic pipe is not restartable. |
<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| Load utility       | Named pipe access module — Loads a large volume of data by writing to a pipe from which Teradata reads. | - Load utilities are faster than INSERT statements through the ODBC driver.  
- Named pipes should be faster than data files because:  
  - As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
  - On Windows, no I/O to an intermediate data file occurs because a pipe is in memory  
- The job is restartable. For details, see "Automatically recovering jobs" in the Designer Guide. | - User must provide the loading script.  
- Cannot use SAP BusinessObjects Data Services parallel processing. |
<p>| MultiLoad, FastLoad, and TPump utilities | File — Loads data by writing to a data file that it passes to the Teradata server. | Load utilities are faster than INSERT or UPSERT statements through the ODBC driver. SAP BusinessObjects Data Services creates the loading script. | Cannot use SAP BusinessObjects Data Services parallel processing |</p>
<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| MultiLoad, FastLoad, and TPump utilities | **Generic named pipe** — Loads a large volume of data by writing to a pipe from which Teradata reads. | • Load utilities are faster than INSERT or UPDATE statements through the ODBC driver.  
• Named pipes are faster than data files because:  
  • As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
  • On Windows, no I/O to an intermediate data file occurs because a pipe is in memory  
  • SAP BusinessObjects Data Services creates the loading script.  | • Cannot use SAP BusinessObjects Data Services parallel processing  
• A job that uses a generic pipe is not restartable. |
<table>
<thead>
<tr>
<th>Bulk loader method</th>
<th>File Option</th>
<th>Advantages</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| MultiLoad, FastLoad, and TPump utilities | Named pipe access module — Loads a large volume of data by writing to a pipe from which Teradata reads.                                                                                                           | • Load utilities are faster than INSERT or UPDATE statements through the ODBC driver.  
  • Named pipes should be faster than data files because:  
    • As soon as SAP BusinessObjects Data Services writes to a pipe, Teradata can read from the pipe.  
    • On Windows, no I/O to an intermediate data file occurs because a pipe is in memory  
    • The job is restartable. For details, see "Automatically recovering jobs" in the Designer Guide.  
    • SAP BusinessObjects Data Services creates the loading script.                                                                                   | Cannot use SAP BusinessObjects Data Services parallel processing.                                                                                   |
| None (use ODBC)                      | Uses Teradata ODBC driver to send separate SQL INSERT statements to load data.                                                                                                                                    | INSERT statements through the ODBC driver are simpler to use than a data file or pipe.                                                                                      | This method does not bulk load data.                                                                                                           |
How SAP BusinessObjects Data Services and Teradata use the file options to load

For both Parallel Transporter and Load Utilities methods, you can choose to use either named pipes or staging data files. Choose from the following file options:

- Data file
- Generic named pipe
- Named pipe access module

Data file

SAP BusinessObjects Data Services runs bulk-loading jobs using a staging data file as follows:

1. It generates staging data file(s) containing data to be loaded into a Teradata table.
2. It generates a loading script to be used by Teradata Parallel Transporter. The script defines read and load operators.
3. If you use Teradata Parallel Transporter, the read operator reads the staging data file, then passes the data to the load operator, which loads data into the Teradata table.

Generic named pipe

SAP BusinessObjects Data Services runs bulk-loading jobs using a generic named pipe as follows:

1. It generates a script that Teradata Parallel Transporter uses to load the database.
2. It creates a pipe to contain the data to load into a Teradata table.

On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format:

/temp/fieldname.dat
On Windows, the name has this format:

```
\\.\pipe\datastorename_ownername_tablename_loadernum.dat
```

3. It executes the loading script. If you use Teradata Parallel Transporter, the script starts Teradata Parallel Transporter and defines read and load operators.

4. It writes data to the pipes.

5. Teradata Parallel Transporter connects to the pipes. Then the read operator reads the named pipe and passes the data to the load operator, which loads the data into the Teradata table.

**Named pipe access module**

SAP BusinessObjects Data Services runs bulk-loading jobs using a named pipe access module as follows:

1. SAP BusinessObjects Data Services generates a script that Teradata Parallel Transporter uses to load the database. The script starts Teradata Parallel Transporter and defines read and load operators.

2. Teradata (Parallel Transporter or non-Parallel Transporter utility) creates named pipes to contain the data to load into a Teradata table.

   On UNIX, the pipe is a FIFO (first in, first out) file that has name of this format:

   ```
   /temp/filename.dat
   ```

   On Windows, the name has this format:

   ```
   \\.\pipe\datastorename_ownername_tablename_loadernum.dat
   ```

3. SAP BusinessObjects Data Services connects to the pipes and writes data to them.

   **Note:**
   When SAP BusinessObjects Data Services tries to connect to the pipes, Teradata Parallel Transporter might not have yet created them. SAP BusinessObjects Data Services tries to connect every second for up to 30 seconds. You can increase the 30-second connection time to up to
100 seconds as follows: In the Designer, select **Tools > Options > Job Server > General** and enter the following:

Section: al_engine

Key: NamedPipeWaitTime

Value: nn

(nn is from 30 to 100)

4. The Teradata Parallel Transporter read operator reads the named pipe and passes the data to the load operator, which loads the data into the Teradata table.

**Using the UPSERT bulk operation**

The purpose of the Teradata UPSERT operation is to update a row, but if no row matches the update, the row is inserted.

In SAP BusinessObjects Data Services, you can only use the Teradata UPSERT operation with the following **Bulk loader** methods:

- MultiLoad
- TPump
- Parallel Transporter

   The UPSERT operation is available only with the following Operators:

   - Stream
   - Update

In SAP BusinessObjects Data Services, you enable UPSERT on the target table editor's **Bulk Loader Options** tab. In the **options** section, for **Bulk Operation**, select **Upsert** (the default is **Insert**).

The additional **Attributes** available when you select **Upsert** include:

- **Ignore missing updates**—Select whether or not to write the missing update rows into the error table. The default is **Yes**.

- **Ignore duplicate updates**—Select whether or not to write an updated duplicate row into the error table. The default is **No**.
After selecting **Upsert**, notice you can also enable the **Use input keys** option on the target editor's **Options** tab.

**Related Topics**
- *Reference Guide: Objects, Target tables*

**Parallel Transporter method**

SAP BusinessObjects Data Services supports Teradata's Parallel Transporter, an ETL tool that consolidates bulk-loading utilities into a single interface.

When you use the Parallel Transporter method, you can leverage SAP BusinessObjects Data Services' powerful parallel processing capabilities to specify a maximum number of files or pipes for the software to use in processing large quantities of data.

With Teradata Parallel Transporter, you can choose from four types of load operators. Each operator processes your data in a slightly different way.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Loads a large amount of data at high speed into an empty table on the Teradata RDBMS. Use this operator when initially loading tables in the data warehouse.</td>
</tr>
<tr>
<td>SQL Inserter</td>
<td>Inserts data into a specified existing table on the Teradata RDBMS. A single SQL session can insert data, while other operators require multiple sessions. Because it uses the fewest RDBMS resources, this is the least intrusive and slowest method of loading data.</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Stream</td>
<td>Allows parallel inserts, updates, and deletes to new or pre-existing Teradata tables. Uses multiple sessions to load data into one or more new or existing tables. Use this operator to maintain tables in the Teradata RDBMS when the system is busy and can provide minimal resources. Unlike Load and Update, which assembles large volumes of rows into blocks then moves those blocks to Teradata RDBMS, Stream loads data one row at a time.</td>
</tr>
<tr>
<td>Update</td>
<td>Allows highly scalable parallel inserts, updates, and deletes of vast amounts of data to new or existing Teradata tables. Use this operator to maintain tables in the data warehouse.</td>
</tr>
</tbody>
</table>

Attributes are different for each operator selection. Teradata Parallel Transporter predefines attribute default values. Note that:

- You can select and modify attribute values.
- Some attribute values are optional and some are required. You must specify a value for each attribute name shown in **bold**.
- Only the Stream and Update operators provide the Ignore Duplicate Inserts attribute and the Upsert option and associated attributes.

With Teradata Parallel Transporter, you can choose to accept the defaults for the following tbuild options:

- **Log directory**
- **Debug all tasks**
- **Trace all tasks**
- **Latency interval (sec)**
- **Checkpoint interval (sec)**
Parallel processing with Teradata Parallel Transporter

SAP BusinessObjects Data Services provides the option for parallel processing when you bulk load data using the Parallel Transporter method.

Using a combination of choices from the Options tab and Bulk Loader Options tab, you specify the number of data files or named pipes as well as the number of Read and Load Operator Instances. The **Number of Loaders** option distributes the workload while Read/Load Operators perform parallel processing.

In the target table Options tab, specify the **Number of Loaders** to control the number of data files or named pipes that SAP BusinessObjects Data Services or Parallel Transporter generates. SAP BusinessObjects Data Services writes data to these files in batches of 999 rows. For example, if you set **Number of Loaders** to 2, the software would generate two data files, writing 999 rows to the first file, then writing the next 999 rows to the second file. If there are more rows to process, the software continues, writing to the first file again, then the second, and so forth.

In the Bulk Loader Options tab, specify the number of instances in the loading scripts. If you set **Number of DataConnector instances** to 2 and **Number of instances** to 2, Parallel Transporter will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The DataConnector (read operator) instances then pass the data to the load operator instances for parallel loading into Teradata.

The Parallel Transporter uses a control file to read staging files or pipes and load data.

**Note:**
Product performance during this type of parallel loading depends on a number of factors such as distribution of incoming data and underlying DBMS capabilities. Under some circumstances, it is possible that specifying parallel loaders can be detrimental to performance. Always test the parallel loading process before moving to production.
To use Parallel Transporter bulk loading with parallel processing

1. In your target table Options tab, specify the **Number of loaders** to control the number of data files or named pipes. SAP BusinessObjects Data Services will write data to these files in batches of 999 rows.

2. In the Bulk Loader Options tab, choose **Parallel Transporter** as your bulk loader.

3. In **File Option**, choose the type of file (**Data File**, **Generic named pipe**, or **Named pipe access module**) to contain the data to bulk load.

4. If you chose **Data File** or **Generic named pipe** in **File Option**, specify the number of read and load instances in the loading scripts.

   If you set Number of instances to 2 (load operators) and Number of DataConnector instances to 2 (read operators), Parallel Transporter will assign the first read operator instance to read one data file and the other instance to read another data file in parallel. The read operator instances then pass the data to the load operator instances for parallel loading into Teradata.

   **Note:**
   If you chose Data File, the value you specify for DataConnector instances (read operators) should be less than or equal to the number of data files.

5. If you chose **Named Pipe Access Module** in **File Option**, specify **Number of instances** (load operators) in the loading scripts.

   Teradata uses the value you specify in Number of loaders to determine the number of read operator instances, as well as the number of named pipes. The DataConnector instances is not applicable when you use Named Pipe Access Module.

   For example, if you set Number of loaders to 2, Parallel Transporter generates two named pipes and assigns one read operator instance to read from one pipe and the other instance to read the other pipe in parallel. If you set Number of instances to 2 (load operators), the read operator instances pass the data to the load operator instances for parallel loading into Teradata.

6. If you specified **Named pipe access module** in **File Option**, you can override the default settings for the following Teradata Access Module parameters: Log Directory, Log Level, Block Size, Fallback Data, File Name, Fallback Data, Directory Path, Signature Checking.
The Teradata Access Module creates a log file to record the load status and writes information to fallback data files. If the job fails, the Teradata Access Module uses the fallback data files to restart the load. The Access Module log file differs from the build log that you specify in the Log directory option in the Teradata datastore.

**Note:**
SAP BusinessObjects Data Services sets the bulk loader directory as the default value for both Log Directory and Fallback Data Directory Path.

For more information about these parameters, see the relevant Teradata tools and utilities documentation.

### Teradata standalone utilities

In addition to the Parallel Transporter interface, SAP BusinessObjects Data Services supports several Teradata utilities that load to and extract from the Teradata database. Each load utility is a separate executable designed to move data into a Teradata database. Choose from the following bulk loader utilities:

<table>
<thead>
<tr>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiLoad</td>
<td>Loads large quantities of data into populated tables. MultiLoad also supports bulk inserts, updates, upserts, and deletions against populated tables.</td>
</tr>
<tr>
<td>FastLoad</td>
<td>Loads unpopulated tables only. Both the client and server environments support FastLoad. Provides a high-performance load (inserts only) to one empty table each session.</td>
</tr>
<tr>
<td>Utility</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>TPump</td>
<td>Uses standard SQL/DML to maintain data in tables. It also contains a method that you can use to specify the percentage of system resources necessary for operations on tables. Allows background maintenance for insert, update, upsert, and delete operations to take place at any time you specify. Used with small data volumes.</td>
</tr>
<tr>
<td>Load</td>
<td>Invokes one of the above utilities (MultiLoad, FastLoad, or TPump) with the interface prior to Data Services version 11.5.1.</td>
</tr>
</tbody>
</table>

## MultiLoad

This procedure describes how to bulk load a table using the Teradata MultiLoad utility.

1. Ensure that your Teradata datastore specifies a value in **TdpId** (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.

2. In the Bulk Loader Options tab of your target table editor, choose **MultiLoad** in the **Bulk loader** drop-down list.

3. In **File Option**, choose the type of file (**Data File**, **Generic named pipe**, or **Named pipe access module**) to contain the data to bulk load. The default is **Data File**.

4. You can specify the following MultiLoad parameters:

<table>
<thead>
<tr>
<th>MultiLoad parameter</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced print output</td>
<td>The default is not to reduce print output.</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>The default is not to encrypt all sessions.</td>
</tr>
</tbody>
</table>
MultiLoad parameter | Short description
--- | ---
Character set | Particular mapping between characters and byte strings (such as ASCII or UTF-8).

For more information about these parameters, see the Teradata MultiLoad Reference.

5. In Attributes, you can usually use the default settings for the following attributes in the MultiLoad script that SAP BusinessObjects Data Services generates.

<table>
<thead>
<tr>
<th>Script attribute</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTable</td>
<td>Table in which Teradata stores the load job status. Specify the restart log table that will maintain the checkpoint information for your MultiLoad job.</td>
</tr>
<tr>
<td>AccountId</td>
<td>Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.</td>
</tr>
<tr>
<td>WorkTable</td>
<td>Teradata uses this table to stage input data.</td>
</tr>
<tr>
<td>ErrorTable1</td>
<td>Teradata uses this table to store errors that it detects during the acquisition phase of the MultiLoad import task.</td>
</tr>
<tr>
<td>ErrorTable2</td>
<td>Teradata uses this table to store errors that it detects during the application phase of the MultiLoad import task.</td>
</tr>
<tr>
<td>Script attribute</td>
<td>Short description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>ErrorLimit</strong></td>
<td>Maximum number of rejected records that Teradata can write to the error table 1 during the acquisition phase of the MultiLoad import task. If used with ErrorPercentage, <strong>ErrorLimit</strong> specifies the number of records that must be sent to the Teradata database before <strong>ErrorPercentage</strong> takes effect.</td>
</tr>
<tr>
<td><strong>ErrorPercentage</strong></td>
<td>Approximate percentage (expressed as an integer) of total records sent so far (<strong>ErrorLimit</strong>) to the Teradata database that the acquisition phase might reject.</td>
</tr>
</tbody>
</table>
| **CheckpointRate** | Interval between checkpoint operations during the acquisition phase. Express this value as either:  
  - The number of rows read from your client system or sent to the Teradata database  
  - An amount of time in minutes |
<p>| <strong>MaxSessions</strong> | Maximum number of MultiLoad sessions for the load job. |
| <strong>MinSessions</strong> | Minimum number of MultiLoad sessions required for the load job to continue. |
| <strong>TenacityHours</strong> | Number of hours that the MultiLoad utility continues trying to logon when the maximum number of load jobs are already running on the Teradata database. |
| <strong>TenacitySleep</strong> | Number of minutes that the MultiLoad utility waits before it retries a logon operation. The default is six minutes. |
| <strong>TableWait</strong> | Number of hours that MultiLoad continues trying to start when one of the target tables is being loaded by some other job. |</p>
<table>
<thead>
<tr>
<th>Script attribute</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmpCheck</td>
<td>Specifies how MultiLoad should respond when an Access Module Processor (AMP) is down.</td>
</tr>
<tr>
<td>IgnoreDuplicate</td>
<td>Select IgnoreDuplicate to not place duplicate rows in error table 2. The default is to load the duplicate rows.</td>
</tr>
</tbody>
</table>

**Note:**
By default, SAP BusinessObjects Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata MultiLoad Reference.

6. If you specified **Data file** in **File Option**, you can increase the **Number of loaders** in the Options tab which increase the number of data files. SAP BusinessObjects Data Services can use parallel processing to write data to multiple data files in batches of 999 rows.

If you specified Generic named pipe or Named pipe access module, Data Services supports only one loader and disables the Number of loaders option.

**Related Topics**
- *Reference Guide: Objects, Target tables (Teradata target table options)*

**FastLoad**

This procedure describes how to bulk load a table using the Teradata FastLoad utility.

1. Ensure that your Teradata datastore specifies a value in **TdpId** (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.

2. In the Bulk Loader Options tab of the target table editor, choose **FastLoad** in the **Bulk loader** drop-down list.
3. In **File Option**, choose the type of file (Data File, Generic named pipe, or Named pipe access module) to contain the data to bulk load.

4. You can specify the following FastLoad parameters:

<table>
<thead>
<tr>
<th>FastLoad parameter</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Encryption</strong></td>
<td>Encrypt data and requests in all sessions used by the job. The default is not to encrypt all sessions.</td>
</tr>
<tr>
<td><strong>Print all requests</strong></td>
<td>Prints every request sent to the Teradata database. The default is not to reduce print output.</td>
</tr>
<tr>
<td><strong>Buffer size</strong></td>
<td>Number of kilobytes for the output buffer that FastLoad uses for messages to the Teradata database. The default is 63 kilobytes which is also the maximum size.</td>
</tr>
<tr>
<td><strong>Character set</strong></td>
<td>Particular mapping between characters and byte strings (such as ASCII or UTF-8).</td>
</tr>
</tbody>
</table>

For more information about these parameters, see the Teradata FastLoad Reference.

5. In **Attributes**, you can usually use the default settings for the following attributes in the FastLoad script that SAP BusinessObjects Data Services generates.

<table>
<thead>
<tr>
<th>Script attribute</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accountld</strong></td>
<td>Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.</td>
</tr>
<tr>
<td>Script attribute</td>
<td>Short description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CheckpointRate</td>
<td>The number of rows sent to the Teradata database between checkpoint operations. The default is not to checkpoint.</td>
</tr>
<tr>
<td>ErrorLimit</td>
<td>Maximum number of rejected records that Teradata can write to the error table 1 while inserting into a FastLoad table.</td>
</tr>
<tr>
<td>ErrorTable1</td>
<td>FastLoad uses this table to store records that were rejected for errors other than unique primary index or duplicate row violation.</td>
</tr>
<tr>
<td>ErrorTable2</td>
<td>FastLoad uses this table to store records that violated the unique primary index constraint.</td>
</tr>
<tr>
<td>MaxSessions</td>
<td>Maximum number of FastLoad sessions for the load job.</td>
</tr>
<tr>
<td>MinSessions</td>
<td>Minimum number of FastLoad sessions required for the load job to continue.</td>
</tr>
<tr>
<td>TenacityHours</td>
<td>Number of hours that the FastLoad utility continues trying to logon when the maximum number of load jobs are already running on the Teradata database.</td>
</tr>
<tr>
<td>TenacitySleep</td>
<td>Number of minutes that the FastLoad utility waits before it retries a logon operation. The default is six minutes.</td>
</tr>
</tbody>
</table>
By default, SAP BusinessObjects Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata FastLoad Reference.

6. If you specified Data file in File Option, you can increase the Number of loaders in the Options tab which increase the number of data files. The software can use parallel processing to write data to multiple data files in batches of 999 rows.

If you specified Generic named pipe or Named pipe access module, SAP BusinessObjects Data Services supports only one loader and disables the Number of loaders option.

**TPump**

This procedure describes how to bulk load a table using the Teradata TPump utility.

1. Ensure that your Teradata datastore specifies a value in Tdpld (Teradata Director Program Identifier). This option identifies the name of the Teradata database to load and is mandatory for bulk loading.

2. In the Bulk Loader Options tab of your target table editor, choose TPump in the Bulk loader drop-down list.

3. In File Option, choose the type of file (Data File, Generic named pipe, or Named pipe access module) to contain the data to bulk load.

4. You can specify the following TPump parameters:

<table>
<thead>
<tr>
<th>FastLoad parameter</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced print output</td>
<td>Reduce the print output of TPump to the minimal information required to determine the success of the job. The default is not to reduce print output.</td>
</tr>
</tbody>
</table>
### FastLoad Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retain Macros</td>
<td>Keep macros that were created during the job run. You can use these macros as predefined macros for subsequent runs of the same job.</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>Encrypt data and requests in all sessions used by the job. The default is not to encrypt all sessions.</td>
</tr>
<tr>
<td>Number of buffers</td>
<td>Number of request buffers that TPump uses for SQL statements to maintain the Teradata database.</td>
</tr>
<tr>
<td>Character set</td>
<td>Particular mapping between characters and byte strings (such as ASCII or UTF-8).</td>
</tr>
<tr>
<td>Configuration file</td>
<td>Configuration file for the TPump job.</td>
</tr>
<tr>
<td>Periodicity value</td>
<td>Controls the rate at which TPump transfers SQL statements to the Teradata database. Value can be between 1 and 600, which specifies the number of periods per minute. The default value is 4 15-second periods per minute.</td>
</tr>
<tr>
<td>Print all requests</td>
<td>Turns on verbose mode which provides additional statistical data in addition to the regular statistics.</td>
</tr>
</tbody>
</table>

For more information about these parameters, see the Teradata Parallel Data Pump Reference.

5. In **Attributes**, you specify SAP BusinessObjects Data Services parameters that correspond to Teradata parameters in TPump commands. You can usually use the default settings for the following parameters in the TPump script that the software generates.
<table>
<thead>
<tr>
<th>TPump command</th>
<th>Data Services parameter in Attributes pane</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>AccountId</td>
<td>Identifier, of up to 30 characters, associated with the user name that will logon to the Teradata database.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>Append</td>
<td>Use the error table specified in ErrorTable. If the table does not exist, TPump creates it. If the structure of the existing error table is not compatible with the error table TPump creates, the job will run into an error when TPump tries to insert or update the error table.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>CheckpointRate</td>
<td>Number of minutes between checkpoint operations. Value must be an unsigned integer from 0 through 60, inclusive. The default is to checkpoint every 15 minutes.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>ErrorLimit</td>
<td>Maximum number of rejected records that TPump can write to the error table while maintaining a table. The default is no limit. If you specify ErrorPercentage, ErrorLimit specifies the number of records that must be sent to the Teradata database before ErrorPercentage takes effect. For example, if ErrorLimit is 100 and ErrorPercentage is 5, 100 records must be sent to the Teradata database before the approximate 5% rejection limit is applied. If only 5 records were rejected when the 100th record is sent, the limit is not exceeded. However, if six records were rejected when the 100th record is sent, TPump stops processing because the limit is exceeded.</td>
</tr>
<tr>
<td>TPump command</td>
<td>Data Services parameter in Attributes pane</td>
<td>Short description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td><strong>ErrorPercentage</strong></td>
<td>Integer value that represents the approximate percent of the total number of records sent to the Teradata Database that might be rejected during the TPump task. You cannot specify this parameter without ErrorLimit.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td><strong>ErrorTable</strong></td>
<td>Name of the table in which TPump stores information about errors and the rejected records.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td><strong>ExecuteMacro</strong></td>
<td>Name of macro to execute. Using predefined macros saves time because TPump does not need to create and drop new macros each time you run a TPump job script.</td>
</tr>
<tr>
<td>DML LABEL</td>
<td><strong>Ignore duplicate inserts</strong></td>
<td>Select <strong>Ignore duplicate inserts</strong> to not place duplicate rows in the error table.</td>
</tr>
<tr>
<td>NAME</td>
<td><strong>JobName</strong></td>
<td>Character string that identifies the name of a job. The maximum length is 16 characters.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td><strong>Latency</strong></td>
<td>Number of seconds that the oldest record resides in the buffer before TPump flushes it to the Teradata database. Value cannot be less than one second. If the <strong>SerializeOn</strong> is not specified, only the current buffer can possibly be stale. If you specify <strong>SerializeOn</strong>, the number of stale buffers can range from zero to the number of sessions.</td>
</tr>
<tr>
<td>TPump command</td>
<td>Data Services parameter in Attributes pane</td>
<td>Short description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Other TPump commands</td>
<td>LogTable</td>
<td>Name of the table to use to write checkpoint information that is required for the safe and automatic restart of a TPump job. The default name has the following format: owner.table_LT</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>MacroDatabase</td>
<td>Name of database to contain any macros TPump uses or builds. The default is to place macros in the same database that contains the TPump target table.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>MaxSessions</td>
<td>Maximum number of sessions for TPump to use to update the Teradata database. SAP BusinessObjects Data Services uses a default of 3.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>MinSessions</td>
<td>Minimum number of sessions for TPump to use to update the Teradata database.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>NoDrop</td>
<td>Do not drop the error table, even if it is empty, at the end of a job. You can use NoDrop with Append to persist the error table, or you can use it alone.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>NoMonitor</td>
<td>Prevents TPump from checking for statement rate changes from, or update status information for, the TPump Monitor.</td>
</tr>
<tr>
<td>IMPORT INFILE</td>
<td>NoStop</td>
<td>Prevents TPump from terminating because of an error associated with a variable-length record.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>Pack</td>
<td>Number of SQL statements to pack into a multiple-statement request. The default is 20 statements per request. The maximum value is 600.</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>PackMaximum</td>
<td>Select PackMaximum to have TPump dynamically determine the number of records to pack within one request. The maximum value is 600.</td>
</tr>
<tr>
<td>TPump command</td>
<td>Data Services parameter in Attributes pane</td>
<td>Short description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>BEGIN LOAD</td>
<td>Rate</td>
<td>Initial maximum rate at which TPump sends SQL statements to the Teradata database. Value must be a positive integer. If unspecified, Rate is unlimited.</td>
</tr>
</tbody>
</table>
|               | Robust                                    | Specifies whether or not to use robust restart logic. Value can be **YES** or **NO**.  
|               |                                           | - **NO** specifies simple restart logic, which cause TPump to begin where the last checkpoint occurred in the job. TPump re-does any processing that occurred after the checkpoint.  
|               |                                           | - **YES** specifies robust restart logic, which you would use for DML statements that change the results when you repeat the operation. Examples of such statements include the following:  
|               |                                           | INSERTs into tables which allow duplicate rows  
|               |                                           | UPDATE foo SET A=A+1... |
| BEGIN LOAD    | SerializeOn                                | Specify a comma separated list of columns to use as the key for rows and guarantee that operations on these rows occur serially. |
| BEGIN LOAD    | TenacityHours                              | Number of hours that the utility tries to log on sessions required to perform the TPump job. The default is four hours. |
| BEGIN LOAD    | TenacitySleep                              | Number of minutes that TPump waits before it retries a logon operation. The default is six minutes. |
Note:
By default, SAP BusinessObjects Data Services uses the bulk loader directory to store the script, data, error, log, and command (bat) files.

For more information about these parameters, see the Teradata Parallel Data Pump Reference.

6. If you specified **Data file** in **File Option**, you can increase the **Number of loaders** in the Options tab which increase the number of data files. The software can use parallel processing to write data to multiple data files in batches of 999 rows.

If you specified Generic named pipe or Named pipe access module, Data Services supports only one loader and disables the Number of loaders option.

Load

To bulk load a Teradata table using the Load utility

1. In the Bulk Loader Options tab of your target table editor, choose **Load** in the **Bulk loader** drop-down list.

2. In **File Option**, choose the type of file (**Data File**, **Generic named pipe**, or **Named pipe access module**) to contain the data to bulk load.

3. Enter a command to be invoked by SAP BusinessObjects Data Services in the Command line text box. For example, `fast load<C:\tera_script\float.ctl`

4. If you chose **Data File** in **File Option**, enter (or browse to) the directory path where you want the software to place your data file.

5. If you chose **Generic named pipe** or **Named pipe access module** in **File Option**, enter the pipe name.
Using Bulk Loading

Bulk loading in Teradata
Other Tuning Techniques
The previous chapters describe the following tuning techniques:

- Maximizing push-down operations
- Using Caches
- Using Parallel Execution
- Distributing Data Flow Execution
- Using Bulk Loading

This section describes other tuning techniques that you can use to adjust performance:

- Source-based performance options
  - Join ordering
  - Minimizing extracted data
  - Using array fetch size
- Target-based performance options
  - Loading method
  - Rows per commit
- Job design performance options
  - Loading only changed data
  - Minimizing data type conversion
  - Minimizing locale conversion
  - Precision in operations

These techniques require that you monitor the change in performance carefully as you make small adjustments.

**Related Topics**

- *Maximizing Push-Down Operations* on page 41
- *Using Caches* on page 65
- *Using Parallel Execution* on page 79
- *Distributing Data Flow Execution* on page 103
- *Using Bulk Loading* on page 125
Source-based performance options

Join ordering

SAP BusinessObjects Data Services reads sources in preparation for the join in an order determined by a "depth first traversal" of a tree starting from the root node. Controlling this order—referred to as the join order in this description—can often have a profound effect on the performance of producing the join result. The join order has no effect on the actual result produced.

Related Topics
- How join ranks affect join order on page 181
- How join ranks are used in outer joins on page 182

How join ranks affect join order

Consider these examples without join ranks set:

<table>
<thead>
<tr>
<th>Table A</th>
<th>Table B</th>
<th>Table C</th>
</tr>
</thead>
</table>

In this join tree, there is only one branch to traverse. Table A is read then joined with Table B. The results from that join are joined with Table C.
In this join tree, the longest branch is traversed first. Table J is read, then joined with Table L. The results are joined with Table M. The results are then joined with Table K.

You can use join ranks to control the order in which the sources are joined. SAP BusinessObjects Data Services joins the child nodes in the order of decreasing join ranks. The join rank associated with a node is the maximum join rank of any node further down the branch.

For example, suppose you are joining four tables in an outer join—tables J through M—and you assign each a different join rank.

In this join tree, the join rank of Table L is considered 3 because the join rank of Table M is greater than the original join rank of Table L. However, the join rank of Table K (5) is still larger than the join rank of Table L. The join order is as follows: Table J is read and joined with Table K. The results are joined with Table L. The result of that join is joined with Table M.

**Note:**
If the value of **Join rank** is 0 and you collected cache statistics for the sources involved in the join, the optimizer determines the join ranks based on the cache statistics.

**How join ranks are used in outer joins**

Join ordering is determined by the type of join you choose to use in SAP BusinessObjects Data Services. There are two types of joins:

- Normal: Created by drawing connection lines from multiple sources in a data flow to the same query and then entering an integer as a **Join rank**
value for each source using source editors. The software orders join operations using these values.

For a normal join between three or more tables, SAP BusinessObjects Data Services internally implements a series of two-way joins. First, it joins the two highest ranked tables together using the join ranking rules outlined above. Next, it joins the result of the first join with the third highest join ranked table. This process continues until all tables are included in the join.

- **Outer Join**: Created by taking a normal join and then using the Outer Join tab of the query to specify the outer and inner sources. The software processes the outer source as if it was assigned the highest join rank.

For an Outer join, the software ignores the Join rank values unless you are using three or more tables and a pair of joins share the same outer join table.

For example, if table A is an outer join to tables B and C, the join rank is calculated by taking the maximum join rank value of the two tables in each outer join pair (A and B and A and C). The pair with the higher join rank becomes the outer loop.

SAP BusinessObjects Data Services implements joins as nested loop joins. The source with the higher join rank or the one specified as an Outer Source in the Outer Join tab becomes an outer loop. If all the ranks are equal or not set (defaults to 0), then the software picks an inner and outer source at random. During job execution, it reads the source in the outer loop one time; and reads the source in the inner loop for each row in the outer loop. Performance improves when fewer rows are read.

The number of output rows for a join depends on the type of join you use in a data flow.

- For a Normal join, the output is limited to the number of rows that meet join conditions.
• For an Outer join, the output is equal to or greater than the number of rows in the source specified as the Outer Source.

Using an Outer join is often preferable with real-time jobs because often you want the whole message passed on whether or not conditions a join looks for in the inner source exist.

In fact, if you do not use an Outer Join to order joins for a message, the software will still process the message as if it has the highest join rank. The message source editor lacks a Join rank option because the software automatically gives it the highest join rank.

Related Topics
• Reference Guide: Transforms, Query

Tips

For a join between two tables, assign a higher join rank value to the larger table and, if possible, cache the smaller table.

For a join between a table and file:
• If the file is small and can be cached, then assign it a lower join rank value and cache it.
• If you cannot cache the file, then assign it a higher join rank value so that it becomes the "outer table" in the join.

For a join between two files, assign a higher rank value to the larger file and, if possible, cache the smaller file.

Minimizing extracted data

The best way to minimize the amount of data extracted from the source systems is to retrieve only the data that has changed since the last time you performed the extraction. This technique is called changed-data capture.

Related Topics
• Designer Guide: Techniques for Capturing Changed Data
Using array fetch size

SAP BusinessObjects Data Services provides an easy way to request and obtain multiple data rows from source databases. The array fetch size feature allows you to retrieve data using fewer requests, thereby significantly reducing traffic on your network. Tuning array fetch size can also reduce the amount of CPU use on the Job Server computer.

The array fetch feature lowers the number of database requests by "fetching" multiple rows (an array) of data with each request. Enter the number of rows to fetch per request in the **Array fetch size** option on any source table editor or SQL transform editor. The default setting is 1000, meaning that with each database request, the software will automatically fetch 1000 rows of data from your source database. The maximum array fetch size that you can specify is 5000 bytes.

It is recommended that you set the array fetch size based on network speed.

**Note:**
Higher array fetch settings will consume more processing memory proportionally to the length of the data in each row and the number of rows in each fetch.

Regardless of the array fetch setting, sources reading columns with an Oracle LONG data type cannot take advantage of this feature. If a selected data column is of type LONG, the array fetch size internally defaults to 1 row per request.

To set the **Array fetch size** parameter

1. Use either a source table editor or an SQL transform editor.
   a. To use a source table editor:
      a. Double-click a source table in the Designer's workspace.
      b. In the **Performance** section of the **Source** tab, enter a number in the **Array fetch size** text box.
   b. To use an SQL transform editor:
      a. Double-click an SQL transform in the Designer's workspace.
      b. In the SQL transform editor, enter a number in the **Array fetch size** text box.
**Array Fetch Size** indicates the number of rows returned in a single fetch call to a source table. The default value is 1000. This value reduces the number of round-trips to the database and can improve performance for table reads.

The Array Fetch Size option does not support long column data types. If the `SELECT` list contains a long column, the software sets the **Array Fetch Size** to 1 and reads one row of data at a time from the database.

2. Click **OK**.

**Tip**

The optimal number for **Array fetch size** depends on the size of your table rows (the number and type of columns involved) as well as the network round-trip time involved in the database requests and responses. If your computing environment is very powerful, (meaning that the computers running the Job Server, related databases, and connections are extremely fast), then try higher values for **Array fetch size** and test the performance of your jobs to find the best setting.

**Target-based performance options**

**Loading method**

You can choose to use regular loading or bulk loading with Data Services. For a regular load, the Parameterized SQL option is automatically selected by SAP BusinessObjects Data Services when it must generate, parse, and compile the statement. By using parameterized SQL, the software can minimize these efforts by using one handle for a set of values instead of one handle per value.

You cannot combine bulk loading with the following options:

- Auto-correct load
- Enable Partitioning
- Number of Loaders
- Full push down to a database
The software automatically selects this optimizer process when the following conditions are met:

- The source and target in a data flow are on the same database
- The database supports the operations in the data flow

If the optimizer pushes down source or target operations, then it ignores the performance options set for sources (Array fetch size, Caching, and Join rank) because SAP BusinessObjects Data Services is not solely processing the data flow.

- Overflow file
- Transactional loading

To improve performance for a regular load (parameterized SQL), you can select the following options from the target table editor. Note that if you use one, you cannot use the others for the same target.

- Enable Partitioning
  Parallel loading option. The number of parallel loads is determined by the number of partitions in the target table.

- Number of Loaders
  Parallel loading option. The number of parallel loads is determined by the number you enter for this option.

**Related Topics**
- *Push-down operations* on page 42
- *Table partitioning* on page 81
- *Using Bulk Loading* on page 125

**Rows per commit**

*Rows per commit* for regular loading defaults to 1000 rows. Setting the *Rows per commit* value significantly affects job performance. Adjust the rows per commit value in the target table editor’s *Options* tab, noting the following rules:

- Do not use negative number signs and other non-numeric characters.
• If you enter nothing or 0, the text box will automatically display 1000.

• If you enter a number larger than 5000, the text box automatically displays 5000, which is the maximum value allowed.

It is recommended that you set rows per commit between 500 and 2000 for best performance. You might also want to calculate a value. To do this, use the following formula:

\[
\text{max\_IO\_size/row\ size (in bytes)}
\]

For most platforms, max\_IO\_size is 64K. For Solaris, max\_IO\_size is 1024K. For a formula to calculate row size, see *Caching sources* on page 67.

Note that even with a value greater than one set for Rows per commit, SAP BusinessObjects Data Services will submit data one row at a time if the following conditions exist:

• You are loading into a database (this scenario does not apply to Oracle databases), and have a column with a LONG datatype attribute.

• You are using an overflow file where the transaction failed. However, once all the rows are loaded successfully, the commit size reverts to the number you entered. In this case, depending on how often a load error happens, performance might be come worse than setting Rows per commit to 1.

### Job design performance options

#### Loading only changed data

Identifying and loading only changed data is called changed-data capture (CDC), which includes only incremental data that has changed since the last refresh cycle. Performance improves because with less data to extract, transform, and load, the job typically takes less time.

**Related Topics**

- *Designer Guide: Techniques for Capturing Changed Data*
Minimizing data type conversion

SAP BusinessObjects Data Services offers very robust and easy-to-use data type conversions via column mappings of different data types. It is recommended that you:

- Avoid unnecessary data conversions.
- Verify that SAP BusinessObjects Data Services is performing the implicit conversions (selected when you drag and drop columns from input to output schemas in the query transform) as expected. This can be done by looking at the warnings generated during job validation.

Minimizing locale conversion

If your jobs do not require the use of different or multi-byte locales, you can increase performance by ensuring that locales are single-byte and not mismatched.

Precision in operations

SAP BusinessObjects Data Services supports the following precision ranges: 0-28, 29-38, 39-67, 68-96. Note that as you decrease precision, performance increases for arithmetic operations and comparison operations. In addition, when processing an arithmetic or boolean operation that includes decimals in different precision ranges, the software converts all to the highest precision range value because it cannot process more than one decimal data type precision range for a single operation. For example, if the software must perform an arithmetic operation for decimals with precision 28 and 38, it converts both to precision 38 before completing the operation.
Index

A
aggregations, pushing to database 43
al_engine process 24
al_jobserver process 24
analyzing trace log files 30
array fetch size 185
auto correct load
  enabling push down 54

B
bulk loading
  DB2 136
  DB2, using import utility 144
  Informix 136
  Oracle 126
  Oracle, conventional-path load 127
  Oracle, direct-path load 127
  Sybase 148
  Teradata 149
  Teradata, Load Utilities 164
  Teradata, Parallel Transporter 160
  using 177
  vs. regular loading 186

caching
  joins 68
  Table_Comparison transform 70
  tables 67
  using a lookup function 69
  caching data 39, 66
caching lookups vs. setting source table as
  outer join 69
changed-data capture, performance, using for 184
conventional-path loading in Oracle 127

D
data
  capturing changes 188
Data Services, optimizing data transformations 42
data transformations
  optimize by caching 28, 39, 66
  optimizing 42
  pushing to source database 47
data type conversion 189
database links
  and datastores 59
  defined 59
  example data flow 61
  importing 60
  software support 60
  tuning performance 63
datastores
  adding database link support 60
  and database links 59
  persistent cache 72
DB2, bulk loading 136
degree of parallelism, enabled for functions 93
degree of parallelism, for data flows 87
Index

E
extracting data
  caching 66
  minimizing 184

F
filtering, pushing to database 43
functions, pushing to database 43

I
import utility for DB2 bulk loading 144
Informix, bulk loading 136
IPC communications
  between data flows 121
  between sub data flows 122

J
join ordering 181
join rank
  outer joins 181
joins, caching 68
joins, pushing to database 43

L
linked datastores
  creating 60
  example data flow 61
  tuning performance 63
Loading 186
logs
  performance monitor 73
lookup function, caching 69

M
Microsoft SQL Server
  bulk loading 132
  minimize data extracted 184
  monitor log files, performance,
    href=":/perf_opt/measuring with 31

O
operational dashboards 34
optimizing data transformations 42
Oracle
  bulk loading 126
  bulk loading, conventional-path load 127
  bulk loading, direct-path loading 127
  parallel loading 129
ordering, pushing to database 43
outer joins
  rank in 181

P
parallel execution 80, 101
  degree of parallelism 87
  Degree Of Parallelism 87, 108
  degree of parallelism, enabled for functions 93
  file multi-threading 98
  for different data flows and work flows 80
  partitioned tables 81
  partitioned tables, creating 83
  partitioned tables, enabling 86
  using table partitioning and DOP 95
  within a single data flow 81, 101
parallel loading in Oracle 129
peer-to-peer communications
  between data flows 121
  between sub data flows 122
performance
  changed-data capture and 188
  data transformations and 47
performance (continued)
improving, DS environment 18, 22
improving, with bulk loading 177
improving, with parallel execution 80, 101
measuring, with DI options 25
measuring, with DS options 34
tuning, with DS job design options 39, 180
tuning, with DS source options 39, 180
tuning, with DS target options 39, 180
performance monitor log 73
Performance Monitor, reading execution statistics 32
persistent cache datastore 72
projection, pushing to database 43
pushing operations to database
  example 45
  logic 42
  overview 42

R
rows
  duplicate, avoiding loading 54
rows per commit 187
rows, retrieving multiple 26, 185

S
server groups
  grid computing 118
sizing tables 67
sources
  retrieving multiple rows 26, 185
SQL
  pushing to database 42
Sybase, bulk loading 148

T
Table_Comparison transform, performance, improving 70
tables
  caching 67
  retrieving multiple rows 26, 185
  sizing 67
Teradata
  bulk loading 149
  bulk loading using Load Utilities 164
  bulk loading using Parallel Transporter 160
  load utilities 165, 168, 171, 177
throughput, improving 39
trace log files, analyzing 30
tuning techniques
  array fetch size 185
  caching data 39, 66
  join ordering 181
  minimize data extracted 184
  minimize data type conversion 189
  rows per commit 187