# DBFastTableCopy User Guide

## Version 2.0

## Overview

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Platforms</td>
<td>3</td>
</tr>
<tr>
<td>Functional Definition</td>
<td>3</td>
</tr>
<tr>
<td>Internals</td>
<td>4</td>
</tr>
<tr>
<td>Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>Limitations</td>
<td>4</td>
</tr>
<tr>
<td>Direct Path Limitations</td>
<td>4</td>
</tr>
<tr>
<td>Limitations</td>
<td>5</td>
</tr>
<tr>
<td>Direct Path Limitations</td>
<td>5</td>
</tr>
<tr>
<td>Supported Oracle Versions</td>
<td>5</td>
</tr>
<tr>
<td>Data Types</td>
<td>5</td>
</tr>
<tr>
<td>No Indexes</td>
<td>6</td>
</tr>
<tr>
<td>No Triggers</td>
<td>6</td>
</tr>
<tr>
<td>No Data Filtration</td>
<td>6</td>
</tr>
<tr>
<td>Columns</td>
<td>6</td>
</tr>
<tr>
<td>LONG and LONG RAW Data Types</td>
<td>6</td>
</tr>
<tr>
<td>Table Types</td>
<td>6</td>
</tr>
<tr>
<td>Table Partitioning</td>
<td>7</td>
</tr>
<tr>
<td>Numeric Limits</td>
<td>7</td>
</tr>
<tr>
<td>DBFastTableCopy Modes</td>
<td>7</td>
</tr>
<tr>
<td>Database-to-Database</td>
<td>8</td>
</tr>
<tr>
<td>Information Only</td>
<td>8</td>
</tr>
<tr>
<td>Database Output Format</td>
<td>8</td>
</tr>
<tr>
<td>Thread Models</td>
<td>9</td>
</tr>
<tr>
<td>Read Parallelism</td>
<td>9</td>
</tr>
<tr>
<td>Write Parallelism</td>
<td>11</td>
</tr>
<tr>
<td>Determining Efficient Parallelism</td>
<td>11</td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>11</td>
</tr>
<tr>
<td>Balance of Read and Write Threads</td>
<td>11</td>
</tr>
<tr>
<td>Buffer Model</td>
<td>11</td>
</tr>
<tr>
<td>Tuning the Buffer Size</td>
<td>11</td>
</tr>
</tbody>
</table>
Overview

DBFastTableCopy Version 2.0 is a standalone utility program that copies the data content of a single database table from a source database to a target database. The advantage of DBFastTableCopy is that it uses the most efficient methods available within Oracle, and the most efficient processing design possible, to minimize the time to copy the data. The purpose of a fast data copy is to minimize downtime on the source database, and to minimize the number of captured-in-flight transactions that must be applied to the target database to synchronize the source and target databases. The executable file for DBFastTableCopy is named TblCopy.

DBFastTableCopy is invoked from the command line using command-line parameters, and it also uses a combination of mandatory as well as optional parameters. Mandatory parameters consist of the source and target database Net8 Oracle listener strings, the output mode, the table name, and schema username/password. You can also specify various optional parameters.

DBFastTableCopy Version 2.0 provides the following new features:

- It is now possible to copy tables that have columns defined after a LONG or a LONG RAW data type.
- Functionality has been added to restrict the records to copy by specifying a where clause to apply to the select statement.
- It is now possible to copy tables using the system account. You can supply the program with the schema that owns the table and specify the system (DBA) account login info.
- DBFastTableCopy and DBCompare executables for Oracle 10g are now included in the installation kit.
- You can now query DBFastTableCopy to obtain version information and to obtain help text that will display all of the command line parameters.
- Kits are now available for Linux platforms. A key is required to run the Linux DBFastTableCopy executable and this key restricts operation to a single system. An automated key generation application is available to obtain Linux keys as needed.

Supported Platforms

DBFastTableCopy is supported and is available as a licensed tar file on the following software and hardware platforms:

- Tru64 UNIX V5.1 and above on Alpha
- HP-UX 11i v2 and above on Integrity
- HP-UX 11i v1 and above on PA-RISC
- SUSE Linux Enterprise Server V10.0 and above on x86_64
- Red Hat Enterprise Linux AS release 3.0 and above on Integrity

See the Installation Instructions section for more information.

Functional Definition

DBFastTableCopy copies the data content of a single database table from a source database to the exact same table on the target database. DBFastTableCopy does not create the table on the target database, and performs no other Data Definition Language (DDL) processing (it does not modify the source or target schema in any way). The source database, target database, table name, and schema username/password must be specified to the program through command-line arguments.
DBFastTableCopy uses Net8 to connect to the Oracle database server instances. Currently, DBFastTableCopy supports Oracle 8i, 9i and 10g databases.

Internals

DBFastTableCopy uses multiple processing threads and multiple buffers to take advantage of as much of system CPU and internal memory as it can. This parallel processing, and many other aspects of the copy process, may be controlled through various program parameters.

DBFastTableCopy also has extensive trace and debug logging capabilities. Log generation has some impact on performance, but can be completely turned off while DBFastTableCopy is doing the hard work of bulk table data copying. Performance statistics are reported at the end of the copy run and have minimal impact to the copy performance. For debugging and tuning, a very detailed trace of program steps, buffer contents, and potential errors can be generated.

Assumptions

- **Source and target schemas identical**

  By default, the source database table and the target database table must have the same name. The columns, data types, lengths, scale, precision, and other data must exactly match. DBFastTableCopy will verify this information before starting the copying and an error is generated if any of these characteristics produce a mismatch. You can override the table name on the target by using the `-t` switch.

- **Experienced DBA**

  HP engineering recommends that DBFastTableCopy users have strong Oracle database administration skills in order to run the utility.

- **Oracle Net8 connection between source and target**

  DBFastTableCopy executes on a single node and runs as an Oracle client to one or two Oracle database server instances. DBFastTableCopy may run locally to either the source or target database, but will be most efficient, all other factors being equal, on the node with the lightest processing load, which is usually the target node. Additionally, the target node is more likely to be state-of-the-art and thus a faster performing platform.

DBFastTableCopy relies on the Oracle Net8 data communication technology to perform cross-platform data translation; there is no data modification within DBFastTableCopy. The following diagram shows how Net8 communication is used by DBFastTableCopy.
Limitations

The following sections discuss the limitations of DBFastTableCopy.

Direct Path Limitations

The following are limitations of direct path load, and thus DBFastTableCopy:

- Referential integrity constraints are not supported.
- Clustered tables are not supported.
- Loading of remote objects is not supported.
- LOBs (BLOBs, CLOBs), objects, or collections are not supported.
- Loading of varray columns is not supported.

Supported Oracle Versions

DBFastTableCopy is currently validated and tested for Oracle 8i, 9i, and 10g versions only. Oracle versions 8.1.6, 8.1.7, 9.2.0.1, 9.2.0.2, 9.2.0.4, 9.2.0.6, and 9.2.0.7, and 10.2.0.1.0 have been run successfully with DBFastTableCopy.

DBFastTableCopy Version 2.0 now supports Oracle 10g and provides executables built for Oracle 10g. These executables will copy Oracle tables between Oracle 10g versions only. The OCI interface has changed between Oracle versions 9i and 10g. Attempting to copy a table using the 10g DBFastTableCopy image from Oracle 9i to 10g will cause an error because of these OCI differences. HP engineering recommends that you use the Oracle 9i image on a source system or an intermediate server, if necessary, to “push” the table to Oracle 10g. Engineering is currently looking into a workaround for this issue.

HP recommends that you use an intermediary server to copy tables from Oracle 8i to 10g. You will need to use an intermediary server running Oracle 9i and run a DBFastTableCopy executable built for Oracle 9i.

Data Types

DBFastTableCopy can copy tables containing the most common data types in Oracle databases. There are a number of data types that DBFastTableCopy cannot copy, and tables containing these data types will not be copied. DBFastTableCopy will detect these types, issue an error message, and halt before beginning a copy. The following table shows the Oracle internal data types, and which types DBFastTableCopy can and cannot copy. Note that these are the Oracle internal data types, and not the external data types that Oracle can present to the application.

<table>
<thead>
<tr>
<th>Internal Data Type</th>
<th>Oracle Type Code</th>
<th>Supported by DBFastTableCopy</th>
<th>Column Formatting in ASCII Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2</td>
<td>1</td>
<td>Yes</td>
<td>char[] (max length 4000)</td>
</tr>
<tr>
<td>NUMBER</td>
<td>2</td>
<td>Yes</td>
<td>char[21]</td>
</tr>
<tr>
<td>LONG</td>
<td>8</td>
<td>Yes</td>
<td>Char[n]</td>
</tr>
<tr>
<td>DATE</td>
<td>12</td>
<td>Yes</td>
<td>char[22]</td>
</tr>
<tr>
<td>RAW</td>
<td>23</td>
<td>Yes</td>
<td>char[n]</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>24</td>
<td>Yes</td>
<td>Char[n]</td>
</tr>
<tr>
<td>CHAR</td>
<td>96</td>
<td>Yes</td>
<td>char[] (max length 2000)</td>
</tr>
<tr>
<td>User Type</td>
<td>108</td>
<td>No</td>
<td>none</td>
</tr>
<tr>
<td>CLOB</td>
<td>112</td>
<td>No</td>
<td>none</td>
</tr>
<tr>
<td>BLOB</td>
<td>113</td>
<td>No</td>
<td>none</td>
</tr>
<tr>
<td>BFILE</td>
<td>114</td>
<td>No</td>
<td>none</td>
</tr>
<tr>
<td>UROWID</td>
<td>208</td>
<td>Yes¹</td>
<td>char[] (max length 4000)</td>
</tr>
<tr>
<td>ROWID</td>
<td>11</td>
<td>Yes¹</td>
<td>char[11]</td>
</tr>
</tbody>
</table>

¹ Not stored in the database, changed when data is moved.
**No Indexes**
The direct load API of DBFastTableCopy does not allow there to be indexes on the target database table. The target database table must not be indexed, and required indexes should be applied only after the data copy is complete. There is no implicit row order maintained by DBFastTableCopy.

**No Triggers**
The target table must not have any triggers enabled.

**No Data Filtration**
DBFastTableCopy copies all rows in the table, and provides absolutely no data filtration, data cleansing, or other data modification processes. Incorporating these processes into the copy would result in very poor performance.

It is now possible in DBFastTableCopy Version 2.0 to copy a specific range of records from the source table. A where clause selection criteria may be specified using the –w switch. DBFastTableCopy can also selectively copy individual table partitions (using the –P or –p switches).

**Columns**
DBFastTableCopy supports tables with up to 1024 columns. See the **Numeric Limits** section for more information.

**LONG and LONG RAW Data Types**
There can be only one LONG or LONG RAW data type in a table. Version 1.0 of DBFastTableCopy required that this field be defined as the last column in the table. In Version 2.0 this restriction has been removed. DBFastTableCopy can now move a table that has one or more columns defined after the LONG or LONG RAW field in the source table. The target table definition may have the exact same column layout as the source table or you may redefine the column order such that the LONG or LONG RAW is specified last. There are no command line switches needed to copy these tables.

DBFastTableCopy relies on fixed-length buffer rows to accelerate the copy process. Variable-length data types are buffered to the maximum size of the specific data type. Because the LONG and LONG RAW data types have a maximum size of 2^31-1, it is not practical to buffer these types to the maximum size. Instead, the user should provide the maximum size as an input option (–l) to DBFastTableCopy. Very large buffers can result, and the number of data rows per buffer (specified with the –C switch) may need to be greatly reduced. If the –l switch is not specified, a default size of 5000 is used.

If a value is used that is not large enough, you will receive an ORA-24345 error message, ‘A truncation or null fetch error occurred.’ This means that the default of 5000 or the –l value specified is not large enough. Specify a larger value for –l. It may also be necessary to change the rows per buffer parameter value (–C).

These data types are not subject to endian conversion. The application is responsible for final interpretation. Verify the data content after copying tables with these data types.

**Table Types**
The most common table types can be copied, but some tables may not be copied or have additional restrictions. The following table shows the Oracle table types and their impact on DBFastTableCopy. Note that these restrictions are in addition to the data types contained in the table columns.
### Oracle Table Type

<table>
<thead>
<tr>
<th>Oracle Table Type</th>
<th>DBFastTableCopy Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonpartitioned</td>
<td>Usually a single read thread. Parallel reads depend on ROWID partitioning, which creates parallel threads for the database files. If a table resides in a tablespace that is spread over multiple data files, then ROWID copy mode may be specified to obtain parallel threads.</td>
</tr>
<tr>
<td>Partitioned</td>
<td>DBFastTableCopy can create parallel read threads by partition, and partitions to be copied may be selected (single or multiple partitions may be selected).</td>
</tr>
<tr>
<td>Clustered</td>
<td>DBFastTableCopy cannot copy clustered tables (this is due to an Oracle restriction that you cannot use Direct Path loading to load clustered tables).</td>
</tr>
</tbody>
</table>

### Table Partitioning

DBFastTableCopy can select records by table partition. Tables that are partitioned may be copied by specific partitions. The target table may not be partitioned, or it may be partitioned in a different way. DBFastTableCopy does not perform any validation to determine if the table will be partitioned on the target in the same manner as the source.

DBFastTableCopy can perform much faster on partitioned tables versus nonpartitioned tables. DBFastTableCopy will copy multiple partitions in multiple parallel threads. It cannot do this with nonpartitioned tables.

### Numeric Limits

The following table lists the numeric limits currently set in DBFastTableCopy. These limits are subject to change, and are currently set at practical limits to reduce memory requirements. Exceeding these limits will usually result in DBFastTableCopy aborting with an error, although some may produce unpredictable results.

<table>
<thead>
<tr>
<th>Limit description</th>
<th>Current Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in copied Oracle table</td>
<td>1024 columns</td>
</tr>
<tr>
<td>Length of Oracle table name</td>
<td>50 characters</td>
</tr>
<tr>
<td>Length of Oracle service path</td>
<td>100 characters</td>
</tr>
<tr>
<td>Maximum number of data buffers</td>
<td>100 buffers</td>
</tr>
<tr>
<td>Maximum number of read and write threads</td>
<td>98 threads</td>
</tr>
<tr>
<td>Maximum length of database SELECT string</td>
<td>2000 characters</td>
</tr>
<tr>
<td>Number of database files of the tablespace that the table may reside in (when using ROWID functionality, the -P R switch)</td>
<td>4096</td>
</tr>
<tr>
<td>Table partition name size</td>
<td>60 characters</td>
</tr>
</tbody>
</table>

### DBFastTableCopy Modes

DBFastTableCopy operates in the following two different modes, which are defined by the output required. The mode of operation is set with the -O parameter switch.

- Database-to-Database
- Information Only
Note that DBFastTableCopy uses Net8 (Oracle listeners) to connect to an Oracle database server.

**Database-to-Database**

DBFastTableCopy copies directly from the source database to the target database. It uses the Oracle Call Interface (OCI), which is the Oracle high-performance API for low-level direct path access to the database.

![Diagram of Database-to-Database copy](image)

Because Direct Path skips steps requiring data conversion, it saves processing time on both reads and writes and is very fast.

The Database-to-Database copy mode can employ parallel processing on both the read of the source database, and the write to the target database. If the table is partitioned or the tablespace in which the table resides is spread over multiple Oracle data files, then this parallel processing is possible.

**Information Only**

The Information Only mode reads and processes the table schema of the database, and using information available from the source database, determines if the database can be copied, and the approximate amount of data that would be copied.

![Diagram of Information Only mode](image)

See the **Example of Output from Information Only Mode** section for a sample of the output provided in the information mode.

**Database Output Format**

Output directly to the target database is done by using the Oracle Direct Path technology. Direct Path is the Oracle high-performance solution for writing large amounts of data to a table. In order to have high performance, Direct Path does not validate the data being written. Instead, it simply formats it into the database table structure and attaches it to the database. Since the data is not validated in any way, this method is suitable only for data that is known to be valid for the table into which it will
be inserted. For the DBFastTableCopy process, the data has just been read from an identical table on the source database, so data validation is not an issue.

The following diagram shows the Direct Path processing in Oracle.

**Thread Models**

For performance reasons, DBFastTableCopy uses a threaded execution model, and will gain significant performance improvements on any server with more than one CPU. The number of threads can be adjusted indirectly through program parameters.

There is always at least one read thread and one write thread. On single-CPU hardware, these threads will time-slice each other, and performance will be approximately the same as that for a non-threaded implementation. With two or more CPUs, the read threads and write threads swap buffers to increase parallelism and improve performance.

**Read Parallelism**

To perform parallel reads against a specific table, the table data must be segmented into similar-sized chunks. There are two approaches for parallel reads available in DBFastTableCopy, which can be selected with the `-P` command-line parameter. Parallel reads are possible for tables that are partitioned, and for tables in tablespaces that are spread over multiple Oracle data files.
- **Using partitioning** (`-P P` switch)

  Tables that are partitioned can be accessed by partition by using the `-P P` parameter. DBFastTableCopy determines the partition names by reading the database system tables. Additionally, you can specify the individual partitions to be read and copied in a comma-delimited list with another command-line parameter, `-p partition-name,partition-name, ...`. You can specify one or more partitions. If the `-p` parameter is not used, DBFastTableCopy will copy the entire table (all partitions). DBFastTableCopy will create one read thread for each partition. If the number of read threads has been constrained by the user (`-R` is specified) to less than the number of partitions, then each read thread will need to sequentially read more than one partition. This will add to program overhead and degrade performance.

- **Using ROWID** (`-P R` switch)

  Use this option for a table in a tablespace that is spread over multiple Oracle data files. ROWID is an implicit Oracle row identifier. DBFastTableCopy can determine ranges of ROWIDs (per data file) by reading system tables. As with partitions, DBFastTableCopy develops selections by ROWID range. There will be one read thread created for each range of ROWIDs on each data file on which the table resides. If the number of read threads is constrained by the user (`-R` is specified), then the read threads will sequentially select multiple ROWID ranges per thread. The maximum
Write Parallelism

Write parallelism is available with a direct path write to the target database. DBFastTableCopy will create one write thread for each read thread created.

Determining Efficient Parallelism

On hardware platforms with more than one CPU, determining the appropriate number of threads and the right mixture of read and write threads requires accounting for various factors, which are discussed in the following sections.

Number of CPUs

The number of CPUs affects the efficiency of the threads, but the number of threads can far exceed the number of CPUs. Remember that the threads spend most of their time in I/O operations. Given enough memory for the buffers, the operating system should be allowed to time-slice the threads as needed.

Balance of Read and Write Threads

Currently, DBFastTableCopy does not attempt to balance the number of read and write threads. Optimal performance occurs when no thread is waiting for a buffer; that is, there are both empty buffers for read threads and full buffers for write threads. DBFastTableCopy will trace if threads are waiting for buffers; the less this condition shows up in the trace, the more efficient the copy is working overall. If read threads are waiting for empty buffers, then decreasing the number of read threads may help. If write threads are waiting for full buffers, then decreasing the number of write threads may help. Use the –R and –W parameters to reduce the numbers of read and write threads, respectively.

Buffer Model

Copying data from the source to the target database requires buffering the data in memory during transfer. The threaded execution model requires multiple buffers (at least two) for the threads to efficiently execute in parallel. The minimum number of buffers created is determined by the number of threads running in parallel; each thread must have a buffer to work with. The current algorithm determines the number of buffers as the total of read threads plus the total of write threads.

Buffers must hold the number of database rows to be buffered, with each row containing the maximum length of data in the row’s columns. This can result in very large buffers for tables with long data rows. All of the buffers must exist simultaneously in virtual memory (or better, physical memory) for processing to be most efficient. As with most data-intensive programs, DBFastTableCopy performs better with more memory. Tuning the row buffering count to make the best use of physical memory is important for performance. See the Performance Tuning Considerations: Buffer Size in Database Rows section for more information.

Tuning the Buffer Size

Buffer size affects the number of reads and writes performed against the database table. Generally, the more rows in the buffer the fewer reads and writes, which results in better performance. The buffer size can be modified with the –C switch, which specifies the number of rows per buffer. As a general
rule, tables with large rows should use a smaller \(-C\) value to achieve better performance. Conversely, tables with smaller row sizes should use a larger \(-C\) value.

**Parameters**

DBFastTableCopy is controlled by using command-line parameters. Some of the parameters are mandatory, and some change the effect of others. The parameters may be specified in any order.

**Mandatory Parameters**

The mandatory parameters listed in the following table are required for DBFastTableCopy to perform useful work; if mandatory parameters are incorrect or missing, DBFastTableCopy will terminate without opening databases or other processing. There are no default values for these parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-S &lt;path&gt;)</td>
<td>Source database path, as specified by Oracle. This path is the Oracle destination database Net8 listener.</td>
<td>(-S tpcc_tst1)</td>
</tr>
<tr>
<td>(-D &lt;path&gt;)</td>
<td>Destination database path, as specified by Oracle. This path is the Oracle source database Net8 listener.</td>
<td>(-D tpcc_tst2)</td>
</tr>
<tr>
<td>(-O &lt;mode&gt;)</td>
<td>Type of output (mode); valid values are: (D) = output to database (I) = output information only</td>
<td>(-O D)</td>
</tr>
<tr>
<td>(-T &lt;table&gt;)</td>
<td>Name of the Oracle database table to copy</td>
<td>(-T ordr)</td>
</tr>
<tr>
<td>(-u &lt;username/ password&gt;)</td>
<td>The schema username and password for both the source and target database. By default, it is the same on both databases. The Oracle user must have privileges to write data and set database parameters on the target table being copied. You may specify a different user on the target. See the following, optional (--d) parameter.</td>
<td>(-u scott/tiger)</td>
</tr>
</tbody>
</table>

**Optional Parameters**

Optional parameters modify the processing in DBFastTableCopy. The following table describes the optional parameters; the default value and behavior is indicated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-C &lt;number&gt;)</td>
<td>Number of rows to copy in a block (buffer). See the <strong>Tuning the Buffer Size</strong> section for information on setting this parameter. The minimum buffer size allowed is 50 rows.</td>
<td>32000</td>
<td>(-C 65000)</td>
</tr>
<tr>
<td>(-R &lt;read thread count&gt;)</td>
<td>The maximum number of read threads to be started. This is provided to limit the number of read threads, beyond the limitations that DBFastTableCopy may encounter due to memory constraints or the number of partitions or row segments in the table.</td>
<td>Read and write threads combined cannot exceed 98</td>
<td>(-R 5)</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default Value</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td><code>-W &lt;write thread count&gt;</code></td>
<td>The maximum number of write threads to create. The actual number of write threads created may be less due to memory constraints.</td>
<td>Read and write threads combined cannot exceed 98</td>
<td></td>
</tr>
<tr>
<td><code>-1 &lt;buffer size&gt;</code></td>
<td>The maximum size, in bytes, of a <code>LONG</code> or <code>LONG RAW</code> data column in the source database; this sets the buffer size for this specific column.</td>
<td>5000</td>
<td><code>-1 50000</code></td>
</tr>
</tbody>
</table>
| `-P <read partitioning>` | The partitioning algorithm for reading the source table, which depends on the type of Oracle table. Valid values are:  
  T = Single-stream read, which turns off parallel reading. If this mode is chosen, the number of read threads is forced to 1. Results in a simple selection statement.  
  P = Select by partition. Can only read partitioned tables, and creates a selection for each partition. If the number of read threads is not limited, a read thread is created for each partition, up to the maximum number of threads supported. With this choice, a list of partitions, one or more (comma separated), may be provided with the `-p` parameter. Using the list, the number of partitions to be copied can be limited. If there is no `-p` parameter specified, `DBFastTableCopy` will copy all partitions in the source table.  
  R = Select by `ROWID` (for nonpartitioned tables). `DBFastTableCopy` develops a list of `ROWID` ranges based on Oracle database files on which the table resides, and uses these ranges to select subsets of the table for parallel reads. As with partitioned reads, `DBFastTableCopy` creates as many read threads as there are ranges, unless limited by the maximum number of threads allowed. | `-P T = single stream (table) read` | `-P P`                                 |
| `-p <partition list>` | Specifies which partitions are to be copied; useful to segregate a copy of inactive partitions from copies of active partitions. Used with `-P P` only. If used with any other `-P` value, this parameter is ignored. Do not use spaces in the list. | `-p ORER_1,ORER_2` | `-p ORER_1,ORER_2`                         |
| `-X` | Prevents writing to the target database table, no load started. Useful only for testing read performance.                                                                                                     | Normal write | `-X`                                      |
| `-d <username/password>` | The schema username and password for the target database only. This allows the table to be owned by another schema on the target database. You must have privileges to write data and set database parameters on the target table being copied. | `None`        | `-d scott/tiger`                          |
| `-t <table>` | The name of the table on the target database. This allows you to copy to a different named table on the target. All characteristics of the table must match exactly. The name must be specified in uppercase. | `None`        | `-t ORDERS2`                              |
The name of the schema that owns the table on the source database. The schema name must be specified in uppercase. This same schema name will be used for the destination if `-c` is not specified.

None

The name of the schema that owns the table on the target database. Only specify this option if the schema name is different from the source. The schema name must be specified in uppercase.

None

The name of a file containing a where clause to apply to the select statement to determine the records you wish to copy. The file is assumed to be in the home directory where DBFastTableCopy is being run. You can prepend a relative directory specification off of `$HOME` if you wish. The where clause should be listed in the file such as `WHERE NAME = 'SMITH'`. Do not end the clause with a semicolon.

None

None

Print out help for DBFastTableCopy.

None

Print out the version of DBFastTableCopy.

None

This argument is only used for Linux versions of DBFastTableCopy. Use this argument to specify the required key for your platform. See the Linux Versions section for more information.

None

This argument specifies the interval at which you want a status message of the number of rows copied, that is, every `N` rows. The default is 1 million.

1,000,000

Trace level for debugging. There are four levels, numbered 0 through 3. Each level incorporates those below it and adds more information. See the Using Trace Levels section for a more detailed look at the trace – debugging output.

-2 -Z 0

-2 -Z 2

Linux Versions

New in the Version 2.0 release of DBFastTableCopy and DBCompare is support for two Linux versions: SUSE (SLES V10.0) on x86_64, and Red Hat Enterprise Linux V3.0 on Integrity.

You need a key to run these versions of Linux. To obtain a key to run either of these versions, please contact your local HP representative.

Once you obtain the integer key for your platform, you have two ways in which to provide it to the program. You may provide the key using the `-k` parameter on the command line, such as `-k 42400424`. Alternatively, you can place the integer key in a file named `.keyfile`. This file must be placed in the home directory of the Linux user account in which you are running DBFastTableCopy.
Performance Tuning Considerations: Buffer Size in Database Rows

Buffer size is limited by the amount of virtual memory available. DBFastTableCopy must be able to allocate at least one buffer in order to run. At least two buffers are required for efficient processing with one read and one write thread. DBFastTableCopy will abort if unable to allocate at least one buffer. The memory requirement is a product of the number of rows per buffer, the length of the database row, and the number of buffers to be used. A large number of read and write threads will result in a large number of buffers. These factors can quickly exhaust available memory.

It is also important to note that the contents of the buffers are extremely dynamic (data is retained only in transition), so using dynamic memory, which must be swapped in and out of the physical memory, can have significant impact on performance. Having very large buffers increases the risk of burdening the operating system with excessive page swaps.

The following illustration is a trend line in buffer size (X axis) in rows versus GB per hour transfer rate. Notable is that after the buffer size reaches about 10,000 rows, little further performance increase is obtained for increased buffer size. This appears, from further testing, to be true for any number of read threads and database tables.

Using Trace Levels

Currently, there are four trace levels available. Each trace level includes all the trace elements of the lower-order levels. That is, trace level 3 contains trace level 1 and trace level 2 as well. The trace level sets the amount of detail that is written to stdout during processing. stdout should always be directed at a file, as opposed to a terminal display, for performance and convenience reasons.

Detailed tracing of DBFastTableCopy can be extremely useful in setting appropriate parameters, and for debugging in case of failure. However, detailed tracing does impact performance and can create very large trace output files. For maximum performance, tracing should be turned off (trace level 0, which is the default).
Trace Level 0

With tracing turned off, only information from startup and shutdown is output to maximize performance during the actual copy. The following information is provided:

At startup:
- Program version and copyright
- Source database path
- Destination database path
- Username for database access
- Table being copied
- Number of records in each buffer
- User set read thread count (if any)
- Trace level
- Number of read threads, write threads
- Number of buffers created
- Incremental number of rows copied

At shutdown:
- Total rows copied
- Resource elapsed time for reading and writing
- Total resource time
- Total bytes copied
- Elapsed time for program
- Performance in bytes/hour and GB/hour copied

The following trace file is typical for trace level 0 (no trace). Without trace, DBFastTableCopy displays the input parameters (which are changed by the program if necessary to make them self-consistent). At the end of the copy, the performance statistics are logged.

Command: TblCopy -O D -S tpcc_tst1 -D tpcc_tst2 -P P -u tpcc/tpcc -T ordr

** TblCopy STARTUP **
Version:
Version 2.0
© Copyright 2006 Hewlett-Packard Development Company, L.P.

============================================================================
Source database: tpcc_tst1
Destination database: tpcc_tst2
Table to be copied: ordr
Source table will be read by partitions
User set max read thread count: 4

Output is DATABASE via direct path
Buffer size in rows: 32000
Maximum LONG or LONG RAW size: 5000
Username: tpcc
============================================================================

MAIN Attempting to allocate 9 buffers of length 5680000
MAIN Malloc of buffer 0, size 5680000
MAIN Malloc of buffer 1, size 5680000
MAIN Malloc of buffer 2, size 5680000
MAIN Malloc of buffer 3, size 5680000
MAIN Malloc of buffer 4, size 5680000
MAIN Malloc of buffer 5, size 5680000
MAIN Malloc of buffer 6, size 5680000
MAIN Malloc of buffer 7, size 5680000
MAIN Malloc of buffer 8, size 5680000

MAIN Number of READ THREADS: 4
MAIN Number of WRITE THREADS: 4

**** 870533 TOTAL rows copied

=============================================================
End Clock time: 1071592679726 msecs

TOTAL ELAPSED TIME:
118055 msecs
0 Hrs, 1 Mins, 58.055 Secs

TOTAL READ TIME: 378110 msecs
TOTAL WRITE TIME: 115808 msecs
TOTAL RESOURCE TIME: 493918 msecs

BYTES TRANSFERRED: 982575324
EFFECTIVE XFER RATE: 29962800000 Bytes/Hour, 29.96280 GBytes/Hour

Trace Level 1

Trace level 1 provides the following additional output:

At startup:
- OCI startup
- OCI connection to source database
- If required, creating the partition or ROWID data for multiple selects
- Schema details of the copied table (columns, data names, sizes, buffering)

At shutdown:
- Freeing of buffers
- Freeing of OCI handles

Thread startup
For each read thread:
- OCI startup
- OCI connection to source database
- Buffer acquisition and release
- Data read elapsed time, and number of rows read

For each write thread:
- Buffer acquisition and release
- Data output elapsed time, and number of rows written

Buffer status for all buffers (state, rows in buffer)

To distinguish the traces from various threads, the trace messages are preceded with a short identifier.
The read threads are identified by R and a two-digit thread number, such as R01. The write threads
are identified by W and the thread number, such as W02. Thread numbers do not overlap; for
example, there cannot be a R03 and a W03. Trace messages from the main program have no
preceding identifier.
The following trace file is typical for trace level 1. This limited trace displays the OCI functions called, and is useful for determining possible problems with the Oracle processing.

Command: TblCopy -O D -S RAC2_SRC -D RAC2_TRG -u sapr3/sapr3 -T ABC_FTREET -Z 1

** TblCopy STARTUP **
Version: Version: 2.0
(C) Copyright 2006 Hewlett-Packard Development Company, L.P.

=============================================================
Source database: RAC2_SRC
Destination database: RAC2_TRG
Table to be copied: ABC_FTREET
Source table will be read single-stream
Read thread count set to 1
Output is DATABASE via direct path
Buffer size in rows: 32000
Maximum LONG or LONG RAW size: 5000
Username: sapr3
Trace flags are 0X01
=============================================================
MAIN OCI Environment created
MAIN OCI Error handle created
MAIN OCI source Service handle created
MAIN OCI source Server handle created
MAIN OCI source session handle created
MAIN OCI Source service attach completed
MAIN Connecting to source database RAC2_SRC as user sapr3 with password *****
MAIN OCI Server set within service
MAIN OCI Username and password set
MAIN Connected to ORACLE source as user: sapr3

MAIN OCI destination Service handle created
MAIN OCI destination Server handle created
MAIN OCI destination session handle created
MAIN OCI Destination service attach completed
MAIN Connecting to destination database RAC2_TRG, user sapr3, password *****
MAIN OCI Dest server set within service for destination
MAIN OCI Username and password set for destination
MAIN Connected to ORACLE destination as user: sapr3

MAIN Loading schema from source database
MAIN OCI descriptor handle allocated
MAIN OCI description read from source table
Table <ABC_FTREET> has 7 columns
MAIN Handle for source column list loaded
MAIN OCI dest descriptor handle allocated
MAIN OCI description read from dest table
MAIN OCI description parameter read
MAIN OCI destination table contains 7 columns
MAIN Handle for destination column list loaded
MAIN Got source column handle for 1
1 Db type: 1 Ext. type: 1, size: 3, buffer: 8, offset: 8, prec: 0, scl: 0, dcsid: 2, NO L, MANDT
MAIN Got destination column handle for 1
MAIN Got source column handle for 2
1 Db type: 1 Ext. type: 1, size: 1, buffer: 8, offset: 16, prec: 0, scl: 0, dcsid: 2, NO L, LANGU
MAIN Got destination column handle for 2
MAIN Got source column handle for 3
19

...
R00 OCI SELECT statement handle created  
R00 Select statement prepared  
R00 Select statement executed  
WD01 Connected to ORACLE destination as user: sapr3  

WD01 OCI DP Column count set to 7 in context  
WD01 Got the column list attribute  
WD01 Validated Column list is OCI_PTYPE_LIST  
End of selected data  
R00 DATA ERROR REPORTED 1403  
End of selected data  
WD01 OCI Direct Path stream buffer size set to 917504 in DP Context  
R00 Freeing OCI READ handles...  
R00 OCI READ Handles freed  
joining 1 tid 5  
WD01 Direct Path prepare completed  

WD01 OCI DP stream created - 0 bytes  
WD01 OCI DP column array handle created  
WD01 OCI Column Array reports 7 columns, 4096 rows  
OCI DP stream loaded  
***Processed 4096 rows of 15282  
OCI DP stream loaded  
***Processed 8192 rows of 15282  
OCI DP stream loaded  
***Processed 12288 rows of 15282  
OCI DP stream loaded  
***Processed 15282 rows of 15282  
WD01 ** DP load finish!! **  
WD01 Freeing output OCI handles...  
WD01 OCI output handles freed  
WD01 ** Direct Path Output thread EXITING **  
**** 15282 TOTAL rows copied  

========================================================================  
End Clock time: 1131373486573 msecs  
TOTAL ELAPSED TIME:  
334 msecs (0 Hrs, 0 Mins, 0.334 Secs)  
TOTAL READ TIME: 144 msecs  
TOTAL WRITE TIME: 230 msecs  
TOTAL RESOURCE TIME: 374 msecs  
BYTES TRANSFERRED: 1732344  
EFFECTIVE XFER RATE:  
18669600000 Bytes/Hour  
18.66960 GBytes/Hour  
========================================================================  
MAIN Closing semaphores  
MAIN ** PROGRAM TERMINATION **  

Trace Level 2  
Trace level 2 provides the output of trace level 1 plus the following information:  

- Selection statements generated during reads  
- Buffer usage by both read and write threads  
- Counts of database rows copied  
- Resource time in milliseconds per read or write
The following example shows the additional data logged by trace level 2 (for space considerations the trace data from trace level 1 is not shown in this listing). This excerpt shows the display of the select SQL that a thread will use to obtain rows of the database to be copied.

Getting selection, type is P
R01 Select SQL generated:
   SELECT * FROM ordr PARTITION (ORDR_1)
R01 OCI SELECT statement handle created
R01 Select statement prepared
R01 Select statement executed
R01 Get next buffer : release -1, count: 0
R01 Getting next buffer from 10...
R01 Acquired buffer 0
R01 Buffer 0 masked
R01 F0>

The following excerpt shows the cyclic process of the read thread(s) obtaining a buffer, reading data to the buffer while the write thread(s) obtains a buffer, and writings the data to the target database:

WD05 Get next buffer : release -1, count: 0
WD05 Getting next buffer from 10...
WD05 Waiting for buffer...
R00 Buffer 4 masked
R00 F4>
R02 F1< 32000
R02 Get next buffer : release 01, count: 32000
R02 Buffer 01 released, elapsed 135 set 1
R02 Getting next buffer from 10...
R02 Acquired buffer 5
WD04 Acquired buffer 1
WD04 - 1>
R03 F3< 32000
R03 Get next buffer : release 03, count: 32000
R03 Buffer 03 released, elapsed 122 set 1
R03 Getting next buffer from 10...
R03 Acquired buffer 6
WD07 Acquired buffer 3
WD07 - 3>
R03 Buffer 6 masked
R03 F6>
R02 Buffer 5 masked
R02 F5>
R00 F4< 32000
R00 Get next buffer : release 04, count: 32000
R00 Buffer 04 released, elapsed 130 set 1
R00 Getting next buffer from 10...
R00 Acquired buffer 7
R00 Buffer 7 masked
R00 F7>
WD05 Acquired buffer 4
WD05 - 4>
WD06 Get next buffer : release 00, count: 0
WD06 Buffer 00 released, elapsed 302 set 2
WD06 Getting next buffer from 10...
WD06 Waiting for buffer...
R03 F6< 32000
R03 Get next buffer : release 06, count: 32000
R03 Buffer 06 released, elapsed 127 set 1
R03 Getting next buffer from 10...
R03 Acquired buffer 8
WD06 Acquired buffer 6
WD06 - 6>
Trace Level 3

Trace level 3 provides the output of trace levels 1 and 2 plus the following information:

- Management of the semaphore providing mutex to the buffers
- Status and data fill of the buffers
- Detail of the select statement generation
- Dump (five rows) of the data buffer on read (ASCII and hex dump)
- Dump (five rows) of the data length and flags by column and row

The following example shows the additional output from a trace level 3 log. For clarity and space considerations, the lower-level trace lines are omitted. There is additional detail in how the selection set for each thread is created. The buffer states and data counts are dumped in table format. The data buffers are dumped in ASCII and hex representation (five rows only). This trace level can assist in evaluating the data copy process and the number of read and write threads needed to balance the copy process well.

Because of the complexity of this trace, it is best analyzed by someone familiar with the internal workings of DBFastTableCopy.

Getting selection, type is P
Search 0: ORDR_1

R03 Select SQL generated:
SELECT * FROM ordr PARTITION (ORDR_1)

R03 OCI SELECT statement handle created

R03 getting swap semaphore...
R03 got it!
R03 Getting next buffer from 10...

R03 Acquired buffer 0

R03 F0< 32000

Dumping buffer 0 - length 162

<additional rows of data omitted for space>

Dumping 5 stats

<additional rows of data omitted for space>

R03 Buffer 00 released, elapsed 118 set 1
R03 Getting next buffer from 10...
Example of Output from Information Only Mode

The following example is a sample of the output from running DBFastTableCopy in Information Only mode:

Command: `TblCopy -O I -S RAC2_SRC -D RAC2_TRG -u sapr3/sapr3 -Z 1 -T ABC_FTREET`

** TblCopy STARTUP **
Version: 2.0
(C) Copyright 2006 Hewlett-Packard Development Company, L.P.
============================================================================
Source database:      RAC2_SRC
Destination database: RAC2_TRG
Table to be copied:   ABC_FTREET
Source table will be read single-stream
Read thread count set to 1
Information output ONLY
Buffer size in rows: 32000
Maximum LONG or LONG RAW size: 5000
Username: sapr3
Trace flags are 0X01
============================================================================
MAIN OCI Environment created
MAIN OCI Error handle created
MAIN OCI source Service handle created
MAIN OCI source Server handle created
MAIN OCI source session handle created
MAIN OCI Source service attach completed
MAIN Connecting to source database RAC2_SRC as user sapr3 with password *****
MAIN OCI Server set within service
MAIN OCI Username and password set
MAIN Connected to ORACLE source as user: sapr3
MAIN Loading schema from source database
MAIN OCI descriptor handle allocated
MAIN OCI description read from source table
MAIN OCI description parameter read
Table <ABC_FTREET> has 7 columns
MAIN Handle for source column list loaded
MAIN Got source column handle for 1
  0 Db type:  1 Ext. type: 96, size: 3, buffer: 3, offset: 3, prec: 0, scl: 0, dcsid: 2, NO L, MANDT
MAIN Got source column handle for 2
  1 Db type:  1 Ext. type: 96, size: 1, buffer: 1, offset: 4, prec: 0, scl: 0, dcsid: 2, NO L, LANGU
1 Db type: 1 Ext. type: 96, size: 1, buffer: 1, offset: 4, prec: 0, scl: 0, dcsid: 2, NO L, LANGU
MAIN Got source column handle for 3

2 Db type: 1 Ext. type: 96, size: 16, buffer: 16, offset: 20, prec: 0, scl: 0, dcsid: 2, NO L, TREE_ID
MAIN Got source column handle for 4

3 Db type: 1 Ext. type: 96, size: 30, buffer: 30, offset: 50, prec: 0, scl: 0, dcsid: 2, NO L, NAME
MAIN Got source column handle for 5

4 Db type: 1 Ext. type: 96, size: 30, buffer: 30, offset: 80, prec: 0, scl: 0, dcsid: 2, NO L, NODETEXT
MAIN Got source column handle for 6

5 Db type: 1 Ext. type: 96, size: 72, buffer: 72, offset: 152, prec: 0, scl: 0, dcsid: 2, NO L, LOG_FUNCTION
MAIN Got source column handle for 7

6 Db type: 1 Ext. type: 96, size: 50, buffer: 50, offset: 202, prec: 0, scl: 0, dcsid: 2, NO L, FUNCT_TEXT

Buffer sizes from schema
TblCopy: 202
Oracle: 202
Oracle255: 202
MAIN Row buffer length: 203

*** AT: Normal termination of information-only run

*** TblCopy NORMAL TERMINATION ***
Installation Instructions

DBFastTableCopy is delivered in the same kit as the DBCompare utility. To install these programs onto your system, follow these steps:

1. Untar the tar file into a temporary directory (``tar -xvpf ...``).
2. Change directory to the `HP-DBFastTableCopy_DBCompare_{version}_{platform}` directory.
3. Run install (`./install`).
4. Answer `Y` or `N` as to whether or not you accept the license agreement.
5. There are two sets of images, a set for Oracle 9i environments, and a set for Oracle 10g environments. Rename the DBCompare and TblCopy executables you require into a directory in your path, such as `/usr/local/bin`. You may require root privilege to accomplish this.
6. Make sure the Oracle `lib` directory is in your library `PATH` (that is, the `SHLIB_PATH` environment variable). The default location is at `$ORACLE_HOME/lib`.
7. Read the DBFastTableCopy and DBCompare user guides that are provided in this kit for instructions on how to use the programs. The documents are provided in PDF format.
8. You need a key to run the Linux version of DBFastTableCopy. To obtain a key, please contact your local HP representative. See the `Linux Versions` section for instructions on how to use the key.
For More Information

- For questions, comments, or suggestions regarding DBFastTableCopy, contact:

  transition-products@hp.com