Overview

DBCompare is a standalone utility program that compares the data content of a single database table residing in two different databases. The primary purpose of DBCompare is to validate that a specific database table has been copied correctly from the source database to the target database; it is possible that it may be used for other compares, within its limitations.

DBCompare Version 2.0 provides the following new features:

- Functionality has been added to restrict the records to compare by specifying a where clause to apply to the select statement.
- It is now possible to compare 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 DBCompare 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.

Supported Platforms

DBCompare is supported and is available as a licensed tar file for 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 more information.

DBCompare currently supports Oracle 9i databases.

Functional Definition

DBCompare compares the data content of a single database table residing in two different databases. DBCompare does not modify the table data in either database in any way. The databases to be compared, the table name, and the schema login information must be specified to the program through command-line arguments. The results of the compare are output to stdout. The most effective way to run DBCompare is to use a small UNIX shell script. See the Sample UNIX Shell Script section for an example of such a script.

Internals

DBCompare uses multiple buffers to take advantage of as much of the CPU and internal memory as it can. The entire contents of the tables to be compared must be read during the compare. The comparison process is essentially a roll up of the Cyclic Redundancy Check (CRC) and checksums for the data buffers.
Assumptions

**Source and target schemas identical**
By default, the database tables to be compared must have exactly the same name. The columns, data types, lengths, and other information must match. DBCompare will check for this, and report an error and abort if there are any differences in the schema. You can override the table name on the target by using the `-t` switch.

**Oracle Net8 connection between source and target**
DBCompare runs on a single node (it is not implemented as a client/server; it runs as an Oracle client to two Oracle database servers). DBCompare 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 often more state-of-the-art and usually a faster performing platform.

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

Limitations

The following sections discuss the limitations of DBCompare.

**Direct Path Limitations**
DBCompare has the following direct path load limitations:
- Triggers are not supported.
- Referential integrity constraints are not supported.
- Clustered tables are not supported.
- Loading of remote objects is not supported.
- LOBs (BLOBs and CLOBs), objects, or collections are not supported.
- Loading of varray columns is not supported.
Supported Oracle Versions
DBCompare Version 1.0 is currently validated and characterized for Oracle versions of 9i only. Oracle versions 9.2.0.1, 9.2.0.3, 9.2.0.4, 9.2.0.6 and 9.2.0.7 have been run successfully with DBCompare.

DBCompare Version 2.0 is now supported on Oracle 10g.

No Indexes
DBCompare does not use indexes. To do so would cause extremely poor performance. To ensure maximum performance, DBCompare does not compare row by row.

Columns
DBCompare can compare tables with up to 1024 columns. See the Probability of Error (Statistical Analysis) section for more information.

Comparison Limits
DBCompare is designed only to determine whether or not the data contents of the two tables compared are identical. The methods used to accomplish this include performing checksums on nonordered data and performing cyclic redundancy checks on ordered data. By performing checksums of the CRCs, the probability of an invalid comparison is greatly reduced. For more information about CRCs and checksums and the probabilities of error (that is, a false positive or false negative), see the Table Comparison Accuracy section.

To perform comparisons in the shortest possible time, DBCompare does not order the data rows in the tables. Nonindexed and unsorted data reads from Oracle tables do not guarantee the order in which the rows are delivered, even if reading the same table in the same database. Therefore, simple row-matching comparisons are not viable, and DBCompare cannot provide a list of matched or unmatched data.

DBCompare cannot pinpoint where in the data a mismatch has occurred; it can only indicate that a mismatch has occurred.

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

<table>
<thead>
<tr>
<th>Limit description</th>
<th>Current Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in 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>Number of data buffers</td>
<td>6 buffers</td>
</tr>
</tbody>
</table>

LONG and LONG RAW buffering
LONG and LONG RAW columns may be compared (there can be only one in a table, and it must be the last column). DBCompare creates a fixed-length buffer for any variable-length data type, using the maximum length of the data as the width of the fixed-length buffer. However, the maximum length of LONG and LONG RAW data types is $2^{31} - 1$, which is too large for fixed-length buffering, and in most cases unnecessary. Therefore, the program accepts an option (-l), which is used to specify the
maximum expected length of the `LONG` or `LONG RAW` data types. Use this option with any table comparison with these data types, unless the default buffer size of 5000 bytes is acceptable.

**Inputs and Outputs: Databases**

DBCompare reads directly from both databases simultaneously. There is no output (except that created for tracing) until both databases have been completely read, and the comparison data is tabulated.

**Parameters**

DBCompare is controlled by using mandatory and optional command-line parameters, which are shown in the following tables.

**Mandatory Parameters**

The following table describes the mandatory parameters. If these mandatory parameters are incorrect or missing, DBCompare 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>-A <code>&lt;path&gt;</code> or -S <code>&lt;path&gt;</code></td>
<td>Database path as specified by Oracle. One of the two databases to be compared. This path is specified as an Oracle destination database net8 listener. There is no significance to the database being either the “A” or “B” database, except for identification.</td>
<td>-A tpcc_tst1</td>
</tr>
<tr>
<td>-B <code>&lt;path&gt;</code> or -D <code>&lt;path&gt;</code></td>
<td>Database path as specified by Oracle. The other database in the comparison. This path is specified as an Oracle destination database net8 listener.</td>
<td>-B tpcc_tst2</td>
</tr>
<tr>
<td>-T <code>&lt;table&gt;</code></td>
<td>Name of the Oracle table to copy</td>
<td>-T ordr</td>
</tr>
<tr>
<td>-u <code>&lt;username/password&gt;</code></td>
<td>The schema username and password for both source and target databases. It will use the same on both databases.</td>
<td>-u scott/tiger</td>
</tr>
</tbody>
</table>

**Optional Parameters**

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>-C <code>&lt;number&gt;</code></td>
<td>Number of rows to buffer in each read. Larger numbers require more memory, but run slightly faster. If the buffers are too large for memory (there will be six that are all the same size), DBCompare will abort with an error. This parameter is checked to ensure that it is larger than 5000, as very small buffers result in very poor performance.</td>
<td>5000</td>
<td>-C 32000</td>
</tr>
<tr>
<td>-l <code>&lt;buffer size&gt;</code></td>
<td>The maximum size, in bytes, of the <code>LONG</code> or <code>LONG RAW</code> data column in the database table; this sets the buffer size for this specific column.</td>
<td>5000</td>
<td>-l 80000</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Default</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>-P &lt;partition&gt;</td>
<td>A partition of a table. Compares only a specific partition of the table. Must be a single partition name; you cannot supply more than one.</td>
<td>None</td>
<td>-P DPART1</td>
</tr>
<tr>
<td>-Z &lt;level&gt;</td>
<td>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.</td>
<td>-Z 0</td>
<td>-Z 2</td>
</tr>
<tr>
<td>-d &lt;username/password&gt;</td>
<td>The schema username and password for the target database only (the database specified by –B or –D). This allows the table to exist in another schema on the target database.</td>
<td>None</td>
<td>-d scott/tiger</td>
</tr>
<tr>
<td>-t &lt;table&gt;</td>
<td>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.</td>
<td>None</td>
<td>-t ORDERS2</td>
</tr>
<tr>
<td>-s &lt;schema&gt;</td>
<td>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.</td>
<td>None</td>
<td>-s SAPR3</td>
</tr>
<tr>
<td>-c &lt;schema&gt;</td>
<td>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.</td>
<td>None</td>
<td>-c NEWSAPR3</td>
</tr>
<tr>
<td>-w &lt;filename&gt;</td>
<td>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 DBCompare 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.</td>
<td>None</td>
<td>-w whclauses/D010S</td>
</tr>
<tr>
<td>-h or -H</td>
<td>Print out help for DBCompare.</td>
<td>None</td>
<td>-H</td>
</tr>
<tr>
<td>-v or -V</td>
<td>Print out the version of DBCompare.</td>
<td>None</td>
<td>-V</td>
</tr>
</tbody>
</table>

**Table Comparison Accuracy**

For performance reasons, DBCompare does not perform a row-matching comparison of the two tables. Instead, DBCompare uses checksums and cyclic redundancy checks to determine the match of the entire table with a limited (but very large) probability of accuracy. To perform a row-matching comparison, the compared tables would need to deliver the data in a predictable and equivalent row order. This could be achieved by either unique indexing or full-row sorting. Either of these techniques would impact performance significantly, and since DBCompare compares the entire contents of tables, issues of buffering the sorts, or very long times to build unique indexes, arise. Instead, DBCompare uses mathematical summation of the data to determine the probability of the data match. The following sections discuss the checksums and cyclic redundancy checks techniques in more detail.

**Checksums**

Checksums are commonly used to determine data equivalency, but they have severe limitations. The most common usage is with file copies and archiving. A checksum is simply the sum of all the elements of the data without carry (that is, no overflow). The larger the checksum, the more likely the
checksum is to be accurate. Checksums suffer from a number of deficiencies, the primary one being that the order of the data being summed is not checked because addition is commutative.

You can improve checksum accuracy by using both addition and multiplication to derive two checksums, both of which must match. While multiplication is also commutative, summation and multiplication of different numbers is much less likely to both arrive at equivalent results. For example:

<table>
<thead>
<tr>
<th></th>
<th>Table A</th>
<th>Table B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different elements</td>
<td>( 7 + 2 = 9 )</td>
<td>( 5 + 4 = 9 )</td>
</tr>
<tr>
<td></td>
<td>( 7 \times 2 = 14 )</td>
<td>( 5 \times 4 = 20 )</td>
</tr>
<tr>
<td>Same elements, different order</td>
<td>( 3 + 5 + 4 = 12 )</td>
<td>( 4 + 5 + 3 = 12 )</td>
</tr>
<tr>
<td></td>
<td>( 3 \times 5 \times 4 = 60 )</td>
<td>( 4 \times 5 \times 3 = 60 )</td>
</tr>
</tbody>
</table>

Cyclic Redundancy Checks

Cyclic redundancy checks (CRCs) are similar to checksums in usage, but greatly improve the probability of accuracy, primarily because they are sensitive to data ordering. The process of developing a CRC is more complex, requiring more compute time but part of the processing can be predeveloped before application to the compared data.

A CRC is a technique for calculating a check code for the contents of a message. The check code is produced by the mathematical process of polynomial division, where an entire message is the dividend, a constant polynomial is the divisor, and the resulting remainder is the CRC. This is a powerful method for detecting data transmission errors that is widely used, accepted, and mandated by the industry. If the polynomial is well-chosen, then any errors in a transmission, even small errors in a large transmission, are very likely to cause a change in the CRC calculated for the transmission, thereby revealing the error. Because the rows of a database table may be viewed as “transmission packets,” where both the value and order of the elements are important, CRCs are appropriate for row data.

The length of the CRC influences the probability of error. The CRCs used in DBCompare are currently 32 bits.

Unfortunately, the absence of ordering of the data rows prevents the use of CRCs for column data summation. Columns are therefore summed with checksums. In addition, the CRCs for all rows are check summed. If the resulting checksums are equivalent, then the table contents are considered equivalent by DBCompare.

Probability of Error (Statistical Analysis)

In the case of the DBCompare program, an error (other than technical errors, which cause the program to abort) is defined as either reporting a “false negative” (stating that the data is not equivalent when in fact it is) or a “false positive” (stating that the data is equivalent when it is not).

To determine the probability of these errors, the error rates for the CRCs of the rows must be determined, and the error rates for the checksums of the columns, and then the probabilities may be combined statistically to arrive at a probability estimate. As might be expected, the probability of error is influenced by the total quantity of data, both the length of the rows and the number of rows.
In addition, most database data exhibits skewed distribution of CRC and checksum values. This is due to the nonrandom nature of the data (character data distribution skews to more common characters, such as “e”, and numeric data often shows a high number of 0 values).

Probability of a CRC Row Error
A CRC row error is defined here as a data error in the row that arrives at the same CRC value as the row without an error. This definition is then usable in calculating the probability of the column checksum errors. A CRC is essentially the reminder of a polynomial division into the data row, treated as an n-scale binary polynomial.

The probability that two different random data rows of any size will generate the same CRC is inversely dependent on the size of the CRC. For DBCompare, this is 32 bits, and the probability is approximately $1 / 2^{32}$. So, for any one data row, there is a $1 / 2^{32}$ chance that an undetected error has occurred. In practice, the probability of an undetected error is much less, because the error is not a randomly-selected bit stream, but both the true data and the error data are from a subset of all possible bit streams. For example, ASCII text in the database limits the range of bytes possible in the field(s), and binary data usually has a strong skewing to bytes that contain zero (0).

These skews reduce the probability that an error will go undetected by limiting the range of possible data row combinations. A burst error (a run of two or more error bits) less than or equal to the length of the CRC polynomial can always be detected. The nature of the copy process generally will generate large numbers of repeating errors (such as field misalignment), because the copy process is iterative and errors repeat in each iteration.

Probability of a Checksum Column Error
Because checksums are order-insensitive, a checksum error is defined as errors in two or more values in the column that cause the same checksum as the same column without errors. The checksums in DBCompare are 32-bit values, but every 32-bit column in the data buffers is check-summed, the effect being a checksum as “wide” as the table rows. In addition, the CRCs of the data rows are checksummed.

A checksum will detect any “single bit” error in which one element in the list is erroneous. A checksum will also detect any error where one checksum is zero.

Statistical summation of error probability
Finally, you need to roll up the statistical probabilities to arrive at the probability of an error in the checksums of the CRCs and columns of the table.

Output of DBCompare
The following example shows the typical output from running DBCompare. This example is abridged to remove redundant lines.

Command: ./DBCompare   -S RAC2_SRC -D RAC2_TRG -u sapr3/sapr3 -T ABC FTREET

** DBCompare  STARTUP **
Version:
    DBCompare Version: 2.0
(C) Copyright 2006 Hewlett-Packard Development Company, L.P.
Loading arguments...
Number of records to buffer: 5000
Maximum LONG or LONG RAW size: 5000
User name: sapr3

Trace flag is 00

Initializing CRC 32 lookup table
OCI Environment created
OCI Error handle created
OCI source Service handle A created
OCI source Server handle A created
OCI source session handle A created
Connecting database A
Source service attach A completed

Connecting to source A database RAC2_SRC as user sapr3 with password *****
Server set within service A
Username and password set
Connected to ORACLE source A as user: sapr3

OCI source Service handle B created
OCI source Server handle B created
OCI source session handle B created
Connecting database B
Source service attach B completed

Connecting to source B database RAC2_TRG as user sapr3 with password *****
Server set within service B
Username and password set
Connected to ORACLE source B as user: sapr3

Selection SQL: SELECT * FROM "ABC_FTREET"

OCI SELECT statement handle created
Select statement A prepared
OCI SELECT statement handle created
Select statement B prepared
Select statement A describe done
Table "ABC_FTREET" in database A has 7 columns
Select statement B describe done

Table "ABC_FTREET" in database B has 7 columns

1 Int. type:  1 Ext. type: 1, size:      3, mask size:      8, offset:     8,
alloc:    40000 name: MANDT
2 Int. type:  1 Ext. type: 1, size:      1, mask size:      8, offset:    16,
alloc:    40000 name: LANGU
3 Int. type:  1 Ext. type: 1, size:     16, mask size:     16, offset:    32,
alloc:    80000 name: TREE_ID
4 Int. type:  1 Ext. type: 1, size:     30, mask size:     32, offset:    64,
alloc:   160000 name: NAME
5 Int. type:  1 Ext. type: 1, size:     30, mask size:     32, offset:    96,
alloc:   160000 name: NODETEXT
6 Int. type:  1 Ext. type: 1, size:     72, mask size:     72, offset:   168,
alloc:   360000 name: LOG_FUNCTION
7 Int. type:  1 Ext. type: 1, size:     50, mask size:     56, offset:   224,
alloc:   280000 name: FUNCT_TEXT
Row buffer length before alignment: 224
Row buffer length: 224 - data 'columns': 56

Data buffers allocated
Select statement A executed
Select statement B executed

DATA MATCHES
*** PROGRAM TERMINATION ***

*** AT: program termination
Allocated buffers freed
OCI Handles freed

If the data does not match, the previous success message is replaced with the following display:

*************************************
MISMATCH!!
*************************************

Using Trace Levels

There are currently four trace levels available with DBCompare. 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. For performance and convenience reasons, stdout should always be directed to a file rather than a terminal display.

Detailed tracing of DBCompare 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:

- Program version
- Both database paths
- Username for database access
- Table being compared
- Number of records in each buffer
- Trace level
- Schema details of the compared table (columns, data names, sizes, buffering)
- Whether tables match or not

See the previous output listing for an example of this trace level.

Trace Level 1

Trace level 1 provides the following additional output:

- OCI startup
- OCI connections to databases
- Freeing up of the buffers
- Freeing up of the OCI handles
- CRC checksums at the end of the run (they should match)
- 32-bit column checksums (they should all match)

The following trace file is typical for trace level 1:
** DBCompare STARTUP **

Version:

DBCompare Version: 2.0

(C) Copyright 2006 Hewlett-Packard Development Company, L.P.

Loading arguments...

Source database A: RAC2_SRC
Source database B: RAC2_TRG
Table to compare: "ABC_FTREET"
Number of records to buffer: 5000
Maximum LONG or LONG RAW size: 5000
User name: sapr3

Trace flag is 01

Initializing CRC 32 lookup table
OCI Environment created
OCI Error handle created
OCI source Service handle A created
OCI source Server handle A created
OCI source session handle A created
Connecting database A
Source service attach A completed

Connecting to source A database RAC2_SRC as user sapr3 with password *****
Server set within service A
Username and password set
Connected to ORACLE source A as user: sapr3

OCI source Service handle B created
OCI source Server handle B created
OCI source session handle B created
Connecting database B
Source service attach B completed

Connecting to source B database RAC2_TRG as user sapr3 with password *****
Server set within service B
Username and password set
Connected to ORACLE source B as user: sapr3

Selection SQL: SELECT * FROM "ABC_FTREET"

OCI SELECT statement handle created
Select statement A prepared
OCI SELECT statement handle created
Select statement B prepared
Select statement A describe done

Table "ABC_FTREET" in database A has 7 columns
Select statement B describe done

Table "ABC_FTREET" in database B has 7 columns

1 Int. type: 1 Ext. type: 1, size: 3, mask size: 8, offset: 8,
alloc: 40000 name: MANDT
2 Int. type: 1 Ext. type: 1, size: 1, mask size: 8, offset: 16,
alloc: 40000 name: LANGU
3 Int. type: 1 Ext. type: 1, size: 16, mask size: 16, offset: 32,
alloc: 80000 name: TREE_ID
4 Int. type: 1 Ext. type: 1, size: 30, mask size: 32, offset: 64,
alloc: 160000 name: NAME
5 Int. type: 1 Ext. type: 1, size: 30, mask size: 32, offset: 96,
alloc: 160000 name: NODETEXT
6 Int. type: 1 Ext. type: 1, size: 72, mask size: 72, offset: 168,
alloc: 360000 name: LOG_FUNCTION
7 Int. type: 1 Ext. type: 1, size: 50, mask size: 56, offset: 224,
alloc: 280000 name: FUNCT_TEXT
Row buffer length before alignment: 224
Row buffer length: 224 - data 'columns': 56

Data buffers allocated
Select statement A executed
Select statement B executed

**Starting processing, msecs: 1132600027755

TOTAL ELAPSED TIME:
444 msecs
0 Hrs, 0 Mins, 0.444 Secs

CRC 32 Checksums: A = 0x843ae1d4 B = 0x843ae1d4

Col 0: Checksums: A = 0x132b209c B = 0x132b209c
Col 1: Checksums: A = 0xc0000000 B = 0xc0000000
Col 2: Checksums: A = 0x00000000 B = 0x00000000
Col 3: Checksums: A = 0x00000000 B = 0x00000000
Col 4: Checksums: A = 0x00000000 B = 0x00000000
Col 5: Checksums: A = 0xfc32e9cc B = 0xfc32e9cc
Col 6: Checksums: A = 0x290dfa1f B = 0x290dfa1f
Col 7: Checksums: A = 0x3b2a832b B = 0x3b2a832b

<similar lines deleted for space>

Col 52: Checksums: A = 0xc0f9dd5f B = 0xc0f9dd5f
Col 53: Checksums: A = 0xa459600d B = 0xa459600d
Col 54: Checksums: A = 0xf22843aa B = 0xf22843aa
Col 55: Checksums: A = 0x59000000 B = 0x59000000

===================================== DATA MATCHES
=====================================

*** PROGRAM TERMINATION ***

*** AT: program termination
Allocated buffers freed
OCI Handles freed

Trace Level 2

Trace level 2 provides the output of trace level 1 plus the following information:
• Status of the Reads of database A
• Status of the Reads of database B
• Running of CRC 32 checksums (these usually do not match because of different row ordering of the table in each database)
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):

Data buffers allocated
Select statement A executed
Select statement B executed
Buffer 1, Buffer 2 initial masked

**Starting processing, msecs: 1079016959785
Read database A, 65530 records (1 - 65530)
Read database B, 65530 records (1 - 65530)
Doing comparison
CRC 32 Checksums: A = 0x588dece4 B = 0x7a5ca360
Read database A, 65530 records (65531 - 131060)
Read database B, 65530 records (65531 - 131060)
Doing comparison
CRC 32 Checksums: A = 0x3850e394 B = 0x0807f320

<similar lines deleted for space>

Read database A, 46420 records (44953581 - 45000000)
Read database B, 46420 records (44953581 - 45000000)
Doing comparison
CRC 32 Checksums: A = 0x843ae1d4 B = 0x843ae1d4

TOTAL ELAPSED TIME:
692196 msecs
0 Hrs, 11 Mins, 32.196 Secs
CRC 32 Checksums: A = 0xea06b040 B = 0xea06b040

Col 0: Checksums: A = 0x132b209c B = 0x132b209c
Col 1: Checksums: A = 0xc0000000 B = 0xc0000000
Col 2: Checksums: A = 0x00000000 B = 0x00000000
Col 3: Checksums: A = 0x00000000 B = 0x00000000
Col 4: Checksums: A = 0x00000000 B = 0x00000000

<similar lines deleted for space>

Col 49: Checksums: A = 0x68702d66 B = 0x68702d66
Col 50: Checksums: A = 0xc0f9dd5f B = 0xc0f9dd5f
Col 51: Checksums: A = 0xa459600d B = 0xa459600d
Col 52: Checksums: A = 0xf22843aa B = 0xf22843aa
Col 53: Checksums: A = 0x59000000 B = 0x59000000

===================================== DATA MATCHES
===================================== DATA MATCHES

*** PROGRAM TERMINATION ***

*** AT: program termination
Allocated buffers freed
OCI Handles freed

Trace Level 3
Trace level 3 provides the output of trace levels 1 and 2 plus a CRC 32 computational table (for up to 256 values).

Sample UNIX Shell Script

The following code is a sample script to execute DBCompare. For clarity, all parameters are hard coded, but normally the script would be most useful if parameters were read from a simple text list.

#!/usr/bin/ksh -x
# add "-x" to above line for trace
### script to run DBCompare on HP-UX comparing two databases
### runs in the Korn shell
echo $ORACLE_SID
date
echo ""
# set the parameters
cmd=""

cmd="${cmd} -A tpcc_tst1"

cmd="${cmd} -B tpcc_tst2"

cmd="${cmd} -u tpcc/tpcc"

cmd="${cmd} -T ordr"

cmd="${cmd} -C 32000"

cmd="${cmd} -l 3000"

# # run the DBCompare
#

echo "Params: ${cmd}"

./DBCompare ${cmd} > dbcompare.log 2>&1
date
mv dbcompare.log dbcompare${FSTAMP}.log
Installation Instructions

DBCompare is delivered together with the DBFastTableCopy utility. To install these programs onto your system:

1. Untar the tar file into a temporary directory (`tar xpvf ...`).
2. Change the 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 DBCompare and DBFastTableCopy user guides for instructions on how to use the programs. Both documents are provided in PDF format.
For More Information

- For questions, comments, or suggestions regarding DBCompare, contact: transition-products@hp.com