Quest® Data Connector for Oracle and Hadoop 1.6

User Guide

If you have any questions regarding your potential use of this material, contact:

Quest Software World Headquarters
LEGAL Dept
5 Polaris Way
Aliso Viejo, CA 92656
email: legal@quest.com

Refer to our Web site (www.quest.com) for regional and international office information.

Trademarks

Quest Data Connector for Oracle and Hadoop 1.6
User Guide
Friday, 12 October 2012

Table of Contents

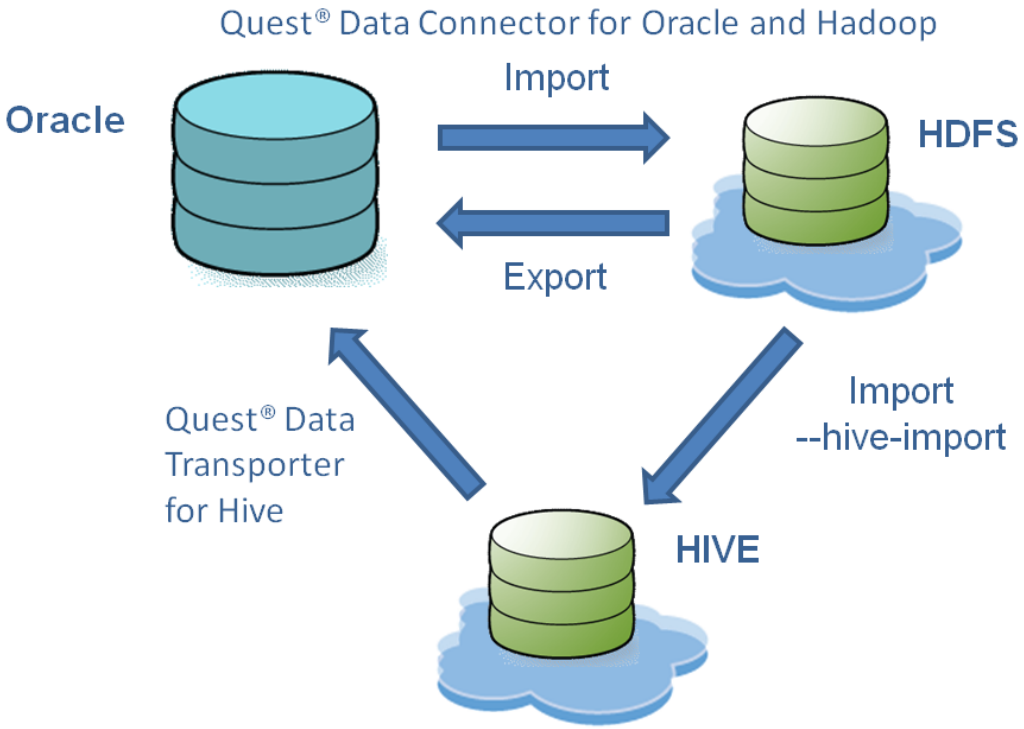
1

About Quest Data Connector for Oracle and Hadoop

What Is Quest Data Connector for Oracle and Hadoop?

Quest Data Connector for Oracle and Hadoop is an optional plugin to Sqoop. It facilitates the movement of data between Oracle and Hadoop.

To Extract From ...	And Import Into...	Do ...
An Oracle Table	HDFS for analysis by Hadoop	Execute Sqoop with Quest Data Connector for Oracle and Hadoop. See "Import Data from Oracle" (page 18) for more information.
An Oracle Table	HDFS for analysis by Hive	Execute Sqoop with Quest Data Connector for Oracle and Hadoop. Add parameter: `--hive-import` See "Import Data from Oracle" (page 18) for more information.
HDFS	An Oracle Table	Execute Sqoop with Quest Data Connector for Oracle and Hadoop. See "Export Data into Oracle" (page 20) for more information.
The Results of a Hive Query	An Oracle Table	See the *Quest Data Transporter for Hive User Guide* for more information.



Quest Data Connector for Oracle and Hadoop Jobs

Quest Data Connector for Oracle and Hadoop inspects each Sqoop job and assumes responsibility for the ones it can perform better than the Oracle manager built into Sqoop.

Quest Data Connector for Oracle and Hadoop accepts responsibility for the following Sqoop Job types:

- **Import** jobs that are **Non-Incremental**.
- **Export** jobs
- Quest Data Connector for Oracle and Hadoop does not accept responsibility for other Sqoop job types. For example Quest Data Connector for Oracle and Hadoop does not accept **eval** jobs etc.

Quest Data Connector for Oracle and Hadoop accepts responsibility for those Sqoop Jobs with the following attributes:

- Oracle-related
- Table-Based — Jobs where the table argument is used and the specified object is a table.

Note: Quest Data Connector for Oracle and Hadoop does not process index-organized tables.

- There are at least 2 mappers — Jobs where the Sqoop command-line does not include:

```
--num-mappers 1
```

How The Oracle Manager Built into Sqoop Works for Imports

The Oracle manager built into Sqoop uses a range-based query for each mapper. Each mapper executes a query of the form:

```
SELECT * FROM sometable WHERE id >= lo AND id < hi
```

The **lo** and **hi** values are based on the number of mappers and the minimum and maximum values of the data in the column the table is being split by.

If no suitable index exists on the table then these queries result in full table-scans within Oracle. Even with a suitable index, multiple mappers may fetch data stored within the same Oracle blocks, resulting in redundant IO calls.

How Quest Data Connector for Oracle and Hadoop Works for Imports

Quest Data Connector for Oracle and Hadoop generates queries for the mappers of the form:

```
SELECT * FROM sometable WHERE
 rowid >= dbms_rowid.rowid_create(1, 893, 1, 279, 0) AND
 rowid <= dbms_rowid.rowid_create(1, 893, 1, 286, 32767)
```

The Quest Data Connector for Oracle and Hadoop queries ensure that:

- No two mappers read data from the same Oracle block. This minimizes redundant IO.
- The table does not require indexes.
- The Sqoop command line does not need to specify a `--split-by` column.

Quest Data Connector for Oracle and Hadoop Exports

Benefits of Quest Data Connector for Oracle and Hadoop:

- **Merge-Export facility** - Update Oracle tables by modifying changed rows AND inserting rows from the HDFS file that did not previously exist in the Oracle table. Quest Data

Connector for Oracle and Hadoop's Merge-Export is unique to Quest Data Connector for Oracle and Hadoop. There is no Sqoop equivalent.

- **Lower impact on the Oracle database** - Update the rows in the Oracle table that have changed, not all rows in the Oracle table. This has performance benefits and reduces the impact of the query on Oracle (for example, the Oracle redo logs).
- **Improved performance** - With partitioned tables, mappers utilize temporary Oracle tables which allow parallel inserts and direct path writes.

2

Download and Install Quest Data Connector for Oracle and Hadoop

Download Quest Data Connector for Oracle and Hadoop

Download Quest Data Connector for Oracle and Hadoop from the Quest Data Connector for Oracle and Hadoop web site: http://www.quest.com/hadoop

Quest Data Connector for Oracle and Hadoop is supplied as a compressed tar archive. The archive contains archives for Quest Data Connector for Oracle and Hadoop and Quest Data Transporter for Hive.

Quest Data Connector for Oracle and Hadoop Archive

Select the archive file appropriate to your use of CDH3 or CDH4. The archive file name is of the form **quest-oraoop-*version*-cdh*version*.tar.gz** where *version* is the numeric identifier of the Quest Data Connector for Oracle and Hadoop release, such as *1.6.0.106* The CDH version is CDH3 or CDH4.

The archive contains the following files:

File	Description
install.sh	A shell script that installs the Quest Data Connector for Oracle and Hadoop files into the Sqoop directory structure.
LICENSE.txt	A text file containing the license for Quest Data Connector for Oracle and Hadoop.
oraoop-*version*.jar	The Java archive (jar) file containing Quest Data Connector for Oracle and Hadoop application code.
oraoop-site-template.xml	The default configuration settings for Quest Data Connector for Oracle and Hadoop jobs running in Hadoop.
oraoopuserguide.pdf	An Adobe PDF version of this Quest Data Connector for Oracle and Hadoop User Guide.
version.txt	A text file containing the version string for this Quest Data Connector for Oracle and Hadoop release.

Install / Upgrade Quest Data Connector for Oracle and Hadoop

Step	Description
Shell login permissions	Login to the shell as a user who has permission to install files into the Sqoop installation directory.
Extract the Quest Data Connector for Oracle and Hadoop archive	Select the Quest Data Connector for Oracle and Hadoop archive appropriate to your use of CDH3 or CDH4. Extract the archive with the command: **`tar xzf quest-oraoop-`***`version`***`-cdh`***`version`***`.tar.gz`** The archive files are extracted into a sub-directory called **oraoop-*version***. **Note:** Ensure the Quest Data Connector for Oracle and Hadoop download archive is saved on the computer where Sqoop is installed.
Location of the Sqoop home directory	Ensure the **SQOOP_HOME** environment variable is set to the absolute path of the Sqoop installation directory.
Location of the Sqoop configuration directory	If the Sqoop site configuration files are located somewhere other than **$SQOOP_HOME/conf**, then set the environment variable **SQOOP_CONF_DIR** to the absolute path of the site configuration directory.
Run the Quest Data Connector for Oracle and Hadoop installer	Install Quest Data Connector for Oracle and Hadoop files into Sqoop directories: 1. Change directory to the sub-directory to which the Quest Data Connector for Oracle and Hadoop files were extracted: **oraoop-*version*** 2. Execute the Quest Data Connector for Oracle and Hadoop shell script file: **install.sh** No arguments are necessary.

Quest Data Connector for Oracle and Hadoop Files In Sqoop Directories

Quest Data Connector for Oracle and Hadoop files are installed into the required locations. Quest Data Connector for Oracle and Hadoop can be installed on any Sqoop client by placing the files listed below in the given locations.

Quest Data Connector for Oracle and Hadoop file	Location
oraoop-*version*.jar	`$SQOOP_HOME/lib`
oraoop-site-template.xml	`$SQOOP_HOME/conf`
oraoop	`$SQOOP_HOME/conf/managers.d`

Ensure The Oracle Database JDBC Driver Is Setup Correctly

You may want to ensure the Oracle Database 11g Release 2 JDBC driver is setup correctly on your system. This driver is required for Sqoop to work with Oracle.

The Oracle Database 11g Release 2 JDBC driver file is **ojdbc6.jar** (3.2Mb).

If this file is not on your system then download it from:
http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

This file should be put into the **$SQOOP_HOME/lib** directory.

Uninstall Quest Data Connector for Oracle and Hadoop

To uninstall the Quest Data Connector for Oracle and Hadoop plug-in, remove all Quest Data Connector for Oracle and Hadoop files from Sqoop directories. See "Quest Data Connector for Oracle and Hadoop Files In Sqoop Directories" (page 11) for more information.

3

Oracle and Quest Data Connector for Oracle and Hadoop

Oracle Roles and Privileges

The Oracle user for Quest Data Connector for Oracle and Hadoop requires the following roles and privileges:

- `create session`

In addition, the user must have the **select any dictionary** privilege or **select_catalog_role** role or all of the following object privileges:

- `select on v_$instance`
- `select on dba_tables`
- `select on dba_tab_columns`
- `select on dba_objects`
- `select on dba_extents`
- `select on dba_segments` — Required for Sqoop imports only
- `select on v_$database` — Required for Sqoop imports only
- `select on v_$parameter` — Required for Sqoop imports only

Note: The user also requires the `alter session` privilege to make use of session tracing functionality. See "Edit oraoop-site.xml" (page 29) for more information. (Section: *oraoop.oracle.session.initialization.statements*)

Additional Oracle Roles And Privileges Required for Export

The Oracle user for Quest Data Connector for Oracle and Hadoop requires:

- Quota on the tablespace in which the Oracle export tables are located.

  An example Oracle command to achieve this is

  alter user *username* quota unlimited on *tablespace*

- The following privileges:

Type of Export	Privileges required

All Export	`create table`
	`select on dba_tab_partitions`
	`select on dba_tab_subpartitions`
	`select on dba_indexes`
	`select on dba_ind_columns`

Insert-Export with a template table into another schema	`select any table`
	`create any table`
	`insert any table`
	`alter any table` (partitioning)

Insert-Export without a template table into another schema	`select,insert on table` (no partitioning) `select,alter on table` (partitioning)
Update-Export into another schema	`select,update on table` (no partitioning) `select,delete,alter,insert on table` (partitioning)
Merge-Export into another schema	`select,insert,update on table` (no partitioning) `select,insert,delete,alter on table` (partitioning)

Supported Data Types

The following Oracle data types are supported by Quest Data Connector for Oracle and Hadoop:

BINARY_DOUBLE
BINARY_FLOAT
BLOB
CHAR
CLOB
DATE
FLOAT
INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH
LONG
NCHAR
NCLOB
NUMBER
NVARCHAR2
RAW
ROWID
TIMESTAMP
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
URITYPE
VARCHAR2

All other Oracle column types are NOT supported. Example Oracle column types NOT supported by Quest Data Connector for Oracle and Hadoop include:

All of the ANY types	BFILE
All of the MEDIA types	LONG RAW
All of the SPATIAL types	MLSLABEL
Any type referred to as UNDEFINED	UROWID
All custom (user-defined) URI types	XMLTYPE

Note: Data types RAW, LONG and LOB (BLOB, CLOB and NCLOB) are supported for Quest Data Connector for Oracle and Hadoop imports. They are not supported for Quest Data Connector for Oracle and Hadoop exports.

4

Execute Sqoop With Quest Data Connector for Oracle and Hadoop

Connect to Oracle / Oracle RAC

The Sqoop *--connect* parameter defines the Oracle instance or Oracle RAC to connect to. It is required with all Sqoop import and export commands.

Quest Data Connector for Oracle and Hadoop expects the associated connection string to be of a specific format dependent on whether the Oracle SID or Service is defined.

```
--connect jdbc:oracle:thin:@OracleServer:OraclePort:OracleSID
```

```
--connect jdbc:oracle:thin:@//OracleServer:OraclePort/OracleService
```

Connect to An Oracle Database Instance

Parameter / Component	Description
`jdbc:oracle:thin`	Quest Data Connector for Oracle and Hadoop requires the connection string starts with **jdbc:oracle**. **Notes:** • The *thin* driver is ojdbc6.jar. See "Ensure The Oracle Database JDBC Driver Is Setup Correctly" (page 12) for more information. • Quest Data Connector for Oracle and Hadoop has been tested with the thin driver however it should work equally well with other drivers such as OCI.
OracleServer	The host name of the Oracle server.
OraclePort	The port to connect to the Oracle server.
OracleSID	The Oracle instance.
OracleService	The Oracle Service.

Notes:

- The Hadoop mappers connect to the Oracle database using a dynamically generated JDBC URL. This is designed to improve performance however it can be disabled by specifying:

  ```
  -D oraoop.jdbc.url.verbatim=true
  ```

Connect to An Oracle RAC

Use the `--connect` parameter as above. The connection string should point to one instance of the Oracle RAC. The listener of the host of this Oracle instance will locate the other instances of the Oracle RAC.

Note: To improve performance, Quest Data Connector for Oracle and Hadoop identifies the active instances of the Oracle RAC and connects each Hadoop mapper to them in a round-robin manner.

If services are defined for this Oracle RAC then use the following parameter to specify the service name:

`-D oraoop.oracle.rac.service.name=`*`ServiceName`*

Parameter / Component	Description
OracleServer :*OraclePort*:*OracleInstance*	Name one instance of the Oracle RAC. **Notes:** • Quest Data Connector for Oracle and Hadoop assumes the same port number for all instances of the Oracle RAC. • The listener of the host of this Oracle instance is used to locate other instances of the Oracle RAC. For more information enter this command on the host command line: `lsnrctl status`
`-D oraoop.oracle.rac.service.name=`*`ServiceName`*	The service to connect to in the Oracle RAC. **Notes:** • A connection is made to all instances of the Oracle RAC associated with the service given by *ServiceName*. • If omitted, a connection is made to all instances of the Oracle RAC. • The listener of the host of this Oracle instance needs to know the *ServiceName* and all instances of the Oracle RAC. For

Parameter / Component	Description
	more information enter this command on the host command line: `lsnrctl status`

Login to The Oracle Instance

Login to the Oracle instance on the Sqoop command line:

```
--connect jdbc:oracle:thin:@OracleServer:OraclePort:OracleInstance --username UserName -P
```

Parameter / Component	Description
`--username` *`UserName`*	The username to login to the Oracle instance (SID).
`-P`	You will be prompted for the password to login to the Oracle instance.

Import Data from Oracle

Execute Sqoop. Following is an example command:

```
$ sqoop import --connect ... --table OracleTableName
```

If Quest Data Connector for Oracle and Hadoop accepts the job then the following text is output:

```
**************************************************************
*** Using Quest Data Connector for Oracle and Hadoop  1.6.x.xxx ***
***             Copyright 2012 Quest Software, Inc.            ***
***                     ALL RIGHTS RESERVED.                   ***
**************************************************************
```

Notes:

- More information is available on the *--connect* parameter. See "Connect to Oracle / Oracle RAC" (page 16) for more information.
- If Java runs out of memory the workaround is to specify each mapper's JVM memory allocation. Add the following parameter for example to allocate 4GB:

  ```
  -Dmapred.child.java.opts=-Xmx4000M
  ```

- An Oracle optimizer hint is included in the SELECT statement by default. See "Edit

oraoop-site.xml" (page 29) for more information.

You can alter the hint on the command line as follows:

```
-Doraoop.import.hint="NO_INDEX(t)"
```

You can turn off the hint on the command line as follows (notice the space between the double quotes):

```
-Doraoop.import.hint=" "
```

Match Hadoop Files to Oracle Table Partitions

```
-Doraoop.chunk.method={ROWID|PARTITION}
```

To import data from a partitioned table in such a way that the resulting HDFS folder structure in Hadoop will match the table's partitions, set the chunk method to PARTITION. The alternative (default) chunk method is ROWID.

Notes:

- For the number of Hadoop files to match the number of Oracle partitions, set the number of mappers to be greater than or equal to the number of partitions.
- If the table is not partitioned then value PARTITION will lead to an error.

Specify The Partitions To Import

```
-Doraoop.import.partitions=PartitionA,PartitionB --table OracleTableName
```

Imports *PartitionA* and *PartitionB* of *OracleTableName*.

Notes:

- You can enclose an individual partition name in double quotes to retain the letter case or if the name has special characters.

  ```
  -Doraoop.import.partitions='"PartitionA",PartitionB' --table
  OracleTableName
  ```

  If the partition name is not double quoted then its name will be automatically converted to upper case, PARTITIONB for above.

  When using double quotes the entire list of partition names must be enclosed in single quotes.

  If the last partition name in the list is double quoted then there must be a comma at the end of the list.

  ```
  -Doraoop.import.partitions='"PartitionA","PartitionB",' --table
  OracleTableName
  ```

- Name each partition to be included. There is no facility to provide a range of partition names.
- There is no facility to define sub partitions. The entire partition is included/excluded as per the filter.

Consistent Read: All Mappers Read From The Same Point In Time

```
-Doraoop.import.consistent.read={true|false}
```

When set to **false** (by default) each mapper runs a select query. This will return potentially inconsistent data if there are a lot of DML operations on the table at the time of import.

Set to **true** to ensure all mappers read from the same point in time. The System Change Number (SCN) is passed down to all mappers, which use the Oracle Flashback Query to query the table as at that SCN.

Notes:

- Values **true** | **false** are case sensitive.
- By default the SCN is taken from V$database. You can specify the SCN in the following command

  ```
  -Doraoop.import.consistent.read.scn=12345
  ```

Export Data into Oracle

Execute Sqoop. Following is an example command:

```
$ sqoop export --connect ... --table OracleTableName --export-dir
/user/username/tablename
```

Quest Data Connector for Oracle and Hadoop accepts all jobs that export data to Oracle. You can verify Quest Data Connector for Oracle and Hadoop is in use by checking the following text is output:

```
**************************************************************
*** Using Quest Data Connector for Oracle and Hadoop  1.6.x.xxx ***
***             Copyright 2012 Quest Software, Inc.            ***
***                    ALL RIGHTS RESERVED.                    ***
**************************************************************
```

Notes:

- *OracleTableName* is the Oracle table the data will export into.
- *OracleTableName* can be in a schema other than that for the connecting user. Prefix the table name with the schema, for example *SchemaName.OracleTableName*.
- Hadoop tables are picked up from the /user/*username*/*tablename* directory.

- The export will fail if the Hadoop file contains any fields of a data type not supported by Quest Data Connector for Oracle and Hadoop. See "Supported Data Types" (page 14) for more information.
- The export will fail if the column definitions in the Hadoop table do not exactly match the column definitions in the Oracle table.
- Quest Data Connector for Oracle and Hadoop indicates if it finds temporary tables that it created more than a day ago that still exist. Usually these tables can be dropped. The only circumstance when these tables should not be dropped is when an Quest Data Connector for Oracle and Hadoop job has been running for more than 24 hours and is still running.
- More information is available on the *--connect* parameter. See "Connect to Oracle / Oracle RAC" (page 16) for more information.

Types of Export

Insert-Export

Appends data to *OracleTableName*. It does not modify existing data in *OracleTableName*.

Insert-Export is the default method, executed in the absence of the `--update-key` parameter. All rows in the HDFS file in user/*UserName*/*TableName* are inserted into *OracleTableName*. No change is made to pre-existing data in *OracleTableName*.

```
$ sqoop export --connect ... --table OracleTableName --export-dir
/user/username/tablename
```

Notes:

- If *OracleTableName* was previously created by Quest Data Connector for Oracle and Hadoop with partitions then this export will create a new partition for the data being inserted.
- When creating *OracleTableName* specify a template. See "Create Oracle Tables" (page 22) for more information.

Update-Export

```
--update-key OBJECT
```

Updates existing rows in *OracleTableName*.

Rows in the HDFS file in user/*UserName*/*TableName* are matched to rows in *OracleTableName* by the *OBJECT* column. Rows that match are copied from the HDFS file to the Oracle table. No action is taken on rows that do not match.

```
$ sqoop export --connect ... --update-key OBJECT --table OracleTableName --
export-dir /user/username/tablename
```

Notes:

- If *OracleTableName* was previously created by Quest Data Connector for Oracle and Hadoop with partitions then this export will create a new partition for the data being inserted. Updated rows will be moved to the new partition that was created for the export.
- For performance reasons it is strongly recommended that where more than a few rows are involved column *OBJECT* be an index column of *OracleTableName*.
- Ensure the column name defined with `--update-key` *OBJECT* is specified in the correct letter case. Sqoop will show an error if the letter case is incorrect.
- It is possible to match rows via multiple columns. See "Match Rows Via Multiple Columns" (page 24) for more information.

Merge-Export

`--update-key` *OBJECT* `-Doraoop.export.merge=true`

Updates existing rows in *OracleTableName*. Copies across rows from the HDFS file that do not exist within the Oracle table.

Rows in the HDFS file in user/*UserName*/*TableName* are matched to rows in *OracleTableName* by the *OBJECT* column. Rows that match are copied from the HDFS file to the Oracle table. Rows in the HDFS file that do not exist in *OracleTableName* are added to *OracleTableName*.

```
$ sqoop export --connect ... --update-key OBJECT -Doraoop.export.merge=true -
-table OracleTableName --export-dir /user/username/tablename
```

Notes:

- Merge-Export is unique to Quest Data Connector for Oracle and Hadoop. It is not a standard Sqoop feature.
- If *OracleTableName* was previously created by Quest Data Connector for Oracle and Hadoop with partitions, then this export will create a new partition for the data being inserted. Updated rows will be moved to the new partition that was created for the export.
- For performance reasons it is strongly recommended that where more than a few rows are involved column *OBJECT* be an index column of *OracleTableName*.
- Ensure the column name defined with `--update-key` *OBJECT* is specified in the correct letter case. Sqoop will show an error if the letter case is incorrect.
- It is possible to match rows via multiple columns. See "Match Rows Via Multiple Columns" (page 24) for more information.

Create Oracle Tables

`-Doraoop.template.table=`*TemplateTableName*

Creates *OracleTableName* by replicating the structure and data types of *TemplateTableName*. *TemplateTableName* is a table that exists in Oracle prior to executing the Sqoop command.

Notes:

- The export will fail if the Hadoop file contains any fields of a data type not supported by Quest Data Connector for Oracle and Hadoop. See "Supported Data Types" (page 14) for more information.
- The export will fail if the column definitions in the Hadoop table do not exactly match the column definitions in the Oracle table.
- This parameter is specific to creating an Oracle table. The export will fail if *OracleTableName* already exists in Oracle.

Example command:

```
$ sqoop export --connect.. --table OracleTableName --export-dir
/user/username/tablename -Doraoop.template.table=TemplateTableName
```

NOLOGGING

```
-Doraoop.nologging=true
```

Assigns the NOLOGGING option to *OracleTableName*.

NOLOGGING may enhance performance but you will be unable to backup the table.

Partitioning

```
-Doraoop.partitioned=true
```

Partitions the table with the following benefits:

- The speed of the export is improved by allowing each mapper to insert data into a separate Oracle table using direct path writes. (An alter table exchange subpartition SQL statement is subsequently executed to swap the data into the export table.)
- You can selectively query or delete the data inserted by each Sqoop export job. For example, you can delete old data by dropping old partitions from the table.

The partition value is the SYSDATE of when Sqoop export job was performed.

The partitioned table created by Quest Data Connector for Oracle and Hadoop includes the following columns that don't exist in the template table:

- **oraoop_export_sysdate** - This is the Oracle SYSDATE when the Sqoop export job was performed. The created table will be partitioned by this column.
- **oraoop_mapper_id** - This is the id of the Hadoop mapper that was used to process the rows from the HDFS file. Each partition is subpartitioned by this column. This column exists merely to facilitate the exchange subpartition mechanism that is performed by each mapper during the export process.
- **oraoop_mapper_row** - A unique row id within the mapper / partition.

Note: If a unique row id is required for the table it can be formed by a combination of **oraoop_export_sysdate**, **oraoop_mapper_id** and **oraoop_mapper_row**.

Match Rows Via Multiple Columns

```
-Doraoop.update.key.extra.columns="ColumnA,ColumnB"
```

Used with Update-Export and Merge-Export to match on more than one column. The first column to be matched on is *--update-key OBJECT*. To match on additional columns, specify those columns on this parameter.

Notes:

- Letter case for the column names on this parameter is not important.
- All columns used for matching should be indexed. The first three items on the index should be *ColumnA*, *ColumnB* and the column specified on *--update-key* - but the order in which the columns are specified is not important.

Storage Clauses

```
-Doraoop.temporary.table.storage.clause="StorageClause"
```

```
-Doraoop.table.storage.clause="StorageClause"
```

Use to customize storage with Oracle clauses as in *TABLESPACE* or *COMPRESS*

-Doraoop.table.storage.clause applies to the export table that is created from *-Doraoop.template.table*. See "Create Oracle Tables" (page 22) for more information. *-Doraoop.temporary.table.storage.clause* applies to all other working tables that are created during the export process and then dropped at the end of the export job.

Kill Quest Data Connector for Oracle and Hadoop Jobs

Use the Hadoop Job Tracker to kill the Sqoop job, just as you would kill any other Map-Reduce job.

```
$ hadoop job -kill jobid
```

To allow an Oracle DBA to kill an Quest Data Connector for Oracle and Hadoop job (via killing the sessions in Oracle) you need to prevent Map-Reduce from re-attempting failed jobs. This is done via the following Sqoop command-line switch:

```
-D mapred.map.max.attempts=1
```

This sends instructions similar to the following to the console:

```
10/08/13 14:35:24 INFO oraOop.OraOopManagerFactory:
This Quest Data Connector for Oracle and Hadoop job can be killed via Oracle
by executing the following statement:

      begin
```

```
   for row in (select sid,serial# from v$session where
module='OraOop' and action='import 20100813143524EST') loop
   execute immediate 'alter system kill session ''' || row.sid ||
',' || row.serial# || '''';
   end loop;
end;
```

5

Manage Date And Timestamp Data Types

Import Date And Timestamp Data Types from Oracle

This section lists known differences in the data obtained by performing an Quest Data Connector for Oracle and Hadoop import of an Oracle table versus a native Sqoop import of the same table.

Quest Data Connector for Oracle and Hadoop Does Not Apply A Time Zone to DATE / TIMESTAMP Data Types

Data stored in a DATE or TIMESTAMP column of an Oracle table is not associated with a time zone. Sqoop without Quest Data Connector for Oracle and Hadoop inappropriately applies time zone information to this data.

Take for example the following timestamp in an Oracle DATE or TIMESTAMP column:
`2am on 3rd October, 2010.`

Request Sqoop without Quest Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. The data is adjusted to Melbourne Daylight Saving Time. The data is imported into Hadoop as:
`3am on 3rd October, 2010.`

Quest Data Connector for Oracle and Hadoop does not apply time zone information to these Oracle data-types. Even from a system located in Melbourne Australia, Quest Data Connector for Oracle and Hadoop ensures the Oracle and Hadoop timestamps match. Quest Data Connector for Oracle and Hadoop correctly imports this timestamp as:
`2am on 3rd October, 2010.`

Note: In order for Quest Data Connector for Oracle and Hadoop to ensure data accuracy, Oracle DATE and TIMESTAMP values must be represented by a String, even when `--as-sequencefile` is used on the Sqoop command-line to produce a binary file in Hadoop.

Quest Data Connector for Oracle and Hadoop Retains Time Zone Information in TIMEZONE Data Types

Data stored in a TIMESTAMP WITH TIME ZONE column of an Oracle table is associated with a time zone. This data consists of two distinct parts: when the event occurred and where the event occurred.

When Sqoop without Quest Data Connector for Oracle and Hadoop is used to import data it converts the timestamp to the time zone of the system running Sqoop and omits the component of the data that specifies where the event occurred.

Take for example the following timestamps (with time zone) in an Oracle TIMESTAMP WITH TIME ZONE column:

```
2:59:00 am on 4th April, 2010. Australia/Melbourne
2:59:00 am on 4th April, 2010. America/New York
```

Request Sqoop without Quest Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. From the data imported into Hadoop we know when the events occurred, assuming we know the Sqoop command was run from a system located in the Australia/Melbourne time zone, but we have lost the information regarding where the event occurred.

```
2010-04-04 02:59:00.0
2010-04-04 16:59:00.0
```

Sqoop with Quest Data Connector for Oracle and Hadoop imports the example timestamps as follows. Quest Data Connector for Oracle and Hadoop retains the time zone portion of the data.

```
2010-04-04 02:59:00.0 Australia/Melbourne
2010-04-04 02:59:00.0 America/New_York
```

Quest Data Connector for Oracle and Hadoop Explicitly States Time Zone for LOCAL TIMEZONE Data Types

Data stored in a TIMESTAMP WITH LOCAL TIME ZONE column of an Oracle table is associated with a time zone. Multiple end-users in differing time zones (locales) will each have that data expressed as a timestamp within their respective locale.

When Sqoop without Quest Data Connector for Oracle and Hadoop is used to import data it converts the timestamp to the time zone of the system running Sqoop and omits the component of the data that specifies location.

Take for example the following two timestamps (with time zone) in an Oracle TIMESTAMP WITH LOCAL TIME ZONE column:

```
2:59:00 am on 4th April, 2010. Australia/Melbourne
2:59:00 am on 4th April, 2010. America/New York
```

Request Sqoop without Quest Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. The timestamps are imported correctly but the local time zone has to be guessed. If multiple systems in different locale were executing the Sqoop import it would be very difficult to diagnose the cause of the data corruption.

```
2010-04-04 02:59:00.0
2010-04-04 16:59:00.0
```

Sqoop with Quest Data Connector for Oracle and Hadoop explicitly states the time zone portion of the data imported into Hadoop. The local time zone is GMT by default. You can set the local time zone with parameter:

```
-Doracle.sessionTimeZone=Australia/Melbourne
```

Quest Data Connector for Oracle and Hadoop would import these two timestamps as:

```
2010-04-04 02:59:00.0 Australia/Melbourne
2010-04-04 16:59:00.0 Australia/Melbourne
```

java.sql.Timestamp

To use Sqoop's handling of date and timestamp data types when importing data from Oracle use the following parameter:

```
-Doraoop.timestamp.string=false
```

Note: Sqoop's handling of date and timestamp data types does not store the timezone. However, some developers may prefer Sqoop's handling as Quest Data Connector for Oracle and Hadoop converts date and timestamp data types to string. This may not work for some developers as the string will require parsing later in the workflow.

Export Date And Timestamp Data Types into Oracle

Ensure the data in the HDFS file fits the required format exactly before using Sqoop to export the data into Oracle.

Notes:

- The Sqoop export command will fail if the data is not in the required format.
- ff = Fractional second
- TZR = Time Zone Region

Oracle Data Type	Required Format of The Data in the HDFS File
DATE	`yyyy-mm-dd hh24:mi:ss`
TIMESTAMP	`yyyy-mm-dd hh24:mi:ss.ff`
TIMESTAMPTZ	`yyyy-mm-dd hh24:mi:ss.ff TZR`
TIMESTAMPLTZ	`yyyy-mm-dd hh24:mi:ss.ff TZR`

6

Configure Quest Data Connector for Oracle and Hadoop

oraoop-site-template.xml

The **oraoop-site-template.xml** file is supplied with Quest Data Connector for Oracle and Hadoop. It contains a number of **ALTER SESSION** statements that are used to initialize the Oracle sessions created by Quest Data Connector for Oracle and Hadoop.

If you need to customize these initializations to your environment then:

1. Find **oraoop-site-template.xml** in the Sqoop configuration directory. See "Quest Data Connector for Oracle and Hadoop Files In Sqoop Directories" (page 11) for more information.
2. Copy **oraoop-site-template.xml** to **oraoop-site.xml**.
3. Edit the **ALTER SESSION** statements in **oraoop-site.xml**.

Edit oraoop-site.xml

oraoop.oracle.session.initialization.statements

The value of this property is a semicolon-delimited list of Oracle SQL statements. These statements are executed, in order, for each Oracle session created by Quest Data Connector for Oracle and Hadoop.

The default statements include:

alter session set time_zone = '{oracle.sessionTimeZone|GMT}';

This statement initializes the timezone of the JDBC client. This ensures that data from columns of type TIMESTAMP WITH LOCAL TIMEZONE are correctly adjusted into the timezone of the client and not kept in the timezone of the Oracle database.

Notes:

- There is an explanation to the text within the curly-braces. See "Expressions in oraoop-site.xml" (page 35) for more information..
- A list of the time zones supported by your Oracle database is available by executing the following query: SELECT TZNAME FROM V$TIMEZONE_NAMES;

alter session disable parallel query;

This statement instructs Oracle to not parallelize SQL statements executed by the Quest Data

Connector for Oracle and Hadoop sessions. This Oracle feature is disabled because the Map/Reduce job launched by Sqoop is the mechanism used for parallelization. It is recommended that you not enable parallel query because it can have an adverse effect the load on the Oracle instance and on the balance between Quest Data Connector for Oracle and Hadoop mappers. Some export operations are performed in parallel where deemed appropriate by Quest Data Connector for Oracle and Hadoop. See "Parallelization" (page 41) for more information.
alter session set "_serial_direct_read"=true; This statement instructs Oracle to bypass the buffer cache. This is used to prevent Oracle from filling its buffers with the data being read by Quest Data Connector for Oracle and Hadoop, therefore diminishing its capacity to cache higher prioritized data. Hence, this statement is intended to minimize Quest Data Connector for Oracle and Hadoop's impact on the immediate future performance of the Oracle database.
--alter session set events '10046 trace name context forever, level 8'; This statement has been commented-out. To allow tracing, remove the comment token **--** from the start of the line.

Notes:

- These statements are placed on separate lines for readability. They do not need to be placed on separate lines.
- A statement can be commented-out via the standard Oracle double-hyphen token: “--”. The comment takes effect until the next semicolon.

oraoop.table.import.where.clause.location

Value	Description

SUBSPLIT (default)	When set to this value, the where clause is applied to each subquery used to retrieve data from the Oracle table.
	
	A Sqoop command like:
	`sqoop import -D oraoop.table.import.where.clause.location=SUBSPLIT --table JUNK --where "owner like 'G%'"`
	
	Generates SQL query of the form:
	`SELECT OWNER,OBJECT_NAME FROM JUNK WHERE ((rowid >= dbms_rowid.rowid_create(1, 113320, 1024, 4223664, 0) AND rowid <= dbms_rowid.rowid_create(1, 113320, 1024, 4223671, 32767))) AND (owner like 'G%') UNION ALL SELECT OWNER,OBJECT_NAME FROM JUNK WHERE ((rowid >= dbms_rowid.rowid_create(1, 113320, 1024, 4223672, 0) AND rowid <= dbms_rowid.rowid_create(1, 113320, 1024, 4223679, 32767))) AND (owner like 'G%')`

SPLIT	When set to this value, the where clause is applied to the entire SQL statement used by each split/mapper.
	
	A Sqoop command like:
	`sqoop import -D oraoop.table.import.where.clause.location=SPLIT --table JUNK --where "rownum <= 10"`
	
	Generates SQL query of the form:
	`SELECT OWNER,OBJECT_NAME FROM (`

Value	Description
	SELECT OWNER,OBJECT_NAME FROM JUNK WHERE ((rowid >= dbms_rowid.rowid_create(1, 113320, 1024, 4223664, 0) AND rowid <= dbms_rowid.rowid_create(1, 113320, 1024, 4223671, 32767))) UNION ALL SELECT OWNER,OBJECT_NAME FROM JUNK WHERE ((rowid >= dbms_rowid.rowid_create(1, 113320, 1024, 4223672, 0) AND rowid <= dbms_rowid.rowid_create(1, 113320, 1024, 4223679, 32767)))) WHERE **rownum <= 10** **Notes:** • In this example, there are up to 10 rows imported per mapper. • The SPLIT clause may result in greater overhead than the SUBSPLIT clause because the UNION statements need to be fully materialized before the data can be streamed to the mappers. However, you may wish to use SPLIT in the case where you want to limit the total number of rows processed by each mapper.

oracle.row.fetch.size

The value of this property is an integer specifying the number of rows the Oracle JDBC driver should fetch in each network round-trip to the database. The default value is 5000.

If you alter this setting, confirmation of the change is displayed in the logs of the mappers during the Map-Reduce job.

oraoop.import.hint

The Oracle optimizer hint is added to the SELECT statement for IMPORT jobs as follows:

```
SELECT /*+ NO_INDEX(t) */ * FROM employees;
```

The default hint is `NO_INDEX(t)`

Notes:

- The hint can be added to the command line. See "Import Data from Oracle" (page 18) for more information.
- See the *Oracle Database Performance Tuning Guide* Using Optimizer Hints for more information on Oracle optimizer hints.
- To turn the hint off, insert a space between the <value> elements.

```
<property>
<name>oraoop.import.hint</name>
<value> </value>
</property>
```

oraoop.oracle.append.values.hint.usage

The value of this property is one of: AUTO / ON / OFF.

Value	Description
AUTO	AUTO is the default value. For Quest Data Connector for Oracle and Hadoop 1.6 AUTO is equivalent to OFF.
ON	During export Quest Data Connector for Oracle and Hadoop uses direct path writes to populate the target Oracle table, bypassing the buffer cache. Oracle only allows a single session to perform direct writes against a specific table at any time, so this has the effect of serializing the writes to the table. This may reduce throughput, especially if the number of mappers is high. However, for databases where DBWR is very busy, or where the IO bandwidth to the underlying table is narrow (table resides on a single disk spindle for instance), then setting **oraoop.oracle.append.values.hint.usage** to ON may reduce the load on the Oracle database and possibly increase throughput.
OFF	During export Quest Data Connector for Oracle and Hadoop does not use the `APPEND_VALUES` Oracle hint.

Note: This parameter is only effective on Oracle 11g Release 2 and above.

mapred.map.tasks.speculative.execution

By default speculative execution is disabled for Quest Data Connector for Oracle and Hadoop. This avoids placing redundant load on the Oracle database.

If Speculative execution is enabled, then Hadoop may initiate multiple mappers to read the same blocks of data, increasing the overall load on the database.

oraoop.block.allocation

This setting determines how Oracle's data-blocks are assigned to Map-Reduce mappers.

Note: Applicable to import. Not applicable to export.

Value	Description
ROUNDROBIN (default)	Each chunk of Oracle blocks is allocated to the mappers in a round-robin manner. This helps prevent one of the mappers from being allocated a large proportion of typically small-sized blocks from the start of Oracle data-files. In doing so it also helps prevent one of the other mappers from being allocated a large proportion of typically larger-sized blocks from the end of the Oracle data-files. Use this method to help ensure all the mappers are allocated a similar amount of work.
RANDOM	The list of Oracle blocks is randomized before being allocated to the mappers via a round-robin approach. This has the benefit of increasing the chance that, at any given instant in time, each mapper is reading from a different Oracle data-file. If the Oracle data-files are located on separate spindles, this should increase the overall IO throughput.
SEQUENTIAL	Each chunk of Oracle blocks is allocated to the mappers sequentially. This produces the tendency for each mapper to sequentially read a large, contiguous proportion of an Oracle data-file. It is unlikely for the performance of this method to exceed that of the round-robin method and it is more likely to allocate a large difference in the work between the mappers. Use of this method is generally not recommended.

oraoop.import.omit.lobs.and.long

This setting can be used to omit all LOB columns (BLOB, CLOB and NCLOB) and LONG column from an Oracle table being imported. This is advantageous in troubleshooting, as it provides a convenient way to exclude all LOB-based data from the import.

oraoop.locations

Note: Applicable to import. Not applicable to export.

By default, four mappers are used for a Sqoop import job. The number of mappers can be altered via the Sqoop `--num-mappers` parameter.

If the data-nodes in your Hadoop cluster have 4 task-slots (that is they are 4-CPU core machines) it is likely for all four mappers to execute on the same machine. Therefore, IO may be concentrated between the Oracle database and a single machine.

This setting allows you to control which DataNodes in your Hadoop cluster each mapper executes on. By assigning each mapper to a separate machine you may improve the overall IO performance for the job. This will also have the side-effect of the imported data being more diluted across the machines in the cluster. (HDFS replication will dilute the data across the cluster anyway.)

Specify the machine names as a comma separated list. The locations are allocated to each of the mappers in a round-robin manner.

If using EC2, specify the internal name of the machines. Here is an example of using this parameter from the Sqoop command-line:

```
$ sqoop import -D oraoop.locations=ip-10-250-23-225.ec2.internal,ip-10-250-107-32.ec2.internal,ip-10-250-207-2.ec2.internal,ip-10-250-27-114.ec2.internal --connect...
```

sqoop.connection.factories

This setting determines behavior if Quest Data Connector for Oracle and Hadoop cannot accept the job. By default Sqoop accepts the jobs that Quest Data Connector for Oracle and Hadoop rejects.

Set the value to `com.quest.oraoop.OraOopManagerFactory` when you want the job to fail if Quest Data Connector for Oracle and Hadoop cannot accept the job.

Expressions in oraoop-site.xml

Text contained within curly-braces { and } are expressions to be evaluated prior to the SQL statement being executed. The expression contains the name of the configuration property optionally followed by a default value to use if the property has not been set. A pipe | character is used to delimit the property name and the default value.

For example:

Situation	Sqoop Command
When this Sqoop command is executed	`$ sqoop import -D oracle.sessionTimeZone =US/Hawaii --connect`
The statement within **oraoop-site.xml**	`alter session set time_ zone = '{ oracle.sessionTimeZone|GMT }';`
Becomes	`alter session set time_ zone = 'US/Hawaii'`
If the **oracle.sessionTimeZone** property had not been	`alter session set time_ zone = 'GMT'`

Situation	Sqoop Command
set, then this statement would use the specified default value and would become:	

Note: The **oracle.sessionTimeZone** property can be specified within the **sqoop-site.xml** file if you want this setting to be used all the time.

7

Troubleshooting Quest Data Connector for Oracle and Hadoop

Disable Quest Data Connector for Oracle and Hadoop

```
-D oraoop.disabled=true
```

For example:

```
$ sqoop import -D oraoop.disabled=true --connect...
```

Quest Data Connector for Oracle and Hadoop does not accept Oracle related jobs while it is disabled.

Verify Quest Data Connector for Oracle and Hadoop is disabled by checking Sqoop stdout (standard output). The stdout should have no reference to Quest Data Connector for Oracle and Hadoop or a reference to indicate Quest Data Connector for Oracle and Hadoop was disabled.

Note: Disabling Quest Data Connector for Oracle and Hadoop via the Sqoop command-line disables Quest Data Connector for Oracle and Hadoop for the Sqoop job. It does not permanently switch off Quest Data Connector for Oracle and Hadoop.

Troubleshooting Import Data

Confirm Sqoop Can Import Oracle Without Quest Data Connector for Oracle and Hadoop

Disable Quest Data Connector for Oracle and Hadoop and import the Oracle data

```
$ sqoop import -D oraoop.disabled=true --connect
jdbc:oracle:thin:@OracleServer:OraclePort:OracleServiceName --username
UserName -P --table TableName --verbose
```

Note: It is a good idea to include `--verbose` on the Sqoop command line to assist you in identifying the cause of the problem.

If Sqoop CAN import the Oracle data while Quest Data Connector for Oracle and Hadoop is disabled
See "Confirm The Imported CSV Data Can Be Correctly Parsed" (page 38) for more information.

If Sqoop CANNOT import the Oracle data while Quest Data Connector for Oracle and Hadoop is disabled

1. Check the Oracle JDBC driver is located in the appropriate locations. See "Install / Upgrade Quest Data Connector for Oracle and Hadoop" (page 11) for more information.
2. Check the Sqoop command-line parameters are correct. For example, you may need to specify a **`--split-by`** argument if the table does not have a primary key.

   Note: Quest Data Connector for Oracle and Hadoop removes the need to specify a `--split-by` argument when the table does not have a primary key.
3. Check all the table column types are supported by the Sqoop Oracle Manager. Use the Sqoop `--columns` argument to omit column types that are not supported:

   For example:

   ```
   $ sqoop import --connect ... --table mytable --columns "COL_DATE,COL_NUMBER"
   ```

Confirm The Imported CSV Data Can Be Correctly Parsed

If you are importing data into Hadoop in CSV format (where `--as-sequencefile` is not on the Sqoop command-line) then check the CSV files can be correctly parsed.

CSV files are the output of the (by default 4) mappers that perform the import. For example, the first file's name of the CUSTOMER table within the HDFS filesystem could be: `/user/hadoop/CUSTOMER/part-m-00000`

One technique for checking the CSV files is to:

1. Import the data into Hive (via the `--hive-import` Sqoop argument)
2. Execute Hive queries against the data in the CSV files and compare the results against an equivalent SQL query executed against the original data in Oracle. For example, you could compare a row count of the data, or compare the sum of a numeric column's data.

Causes of CSV file-format problems may include:

- The CSV field delimiter character ("**,**" by default) is present within the Oracle data.
- New-line characters are present within the Oracle data.

Refer to the *Sqoop User guide*: "Output line formatting arguments" for more information.

Note: Importing the data in binary format (with the Sqoop argument `--as-sequencefile`) is the easiest way to avoid problems with CSV file parsing, although that precludes the subsequent use of Hive.

Confirm Sqoop Can Utilize oraoop-*version*.jar

Import the Oracle data again. This time do not disable Quest Data Connector for Oracle and Hadoop. Turn on verbose.

```
$ sqoop import --connect
jdbc:oracle:thin:@OracleServer:OraclePort:OracleServiceName --username
UserName -P --table TableName --verbose
```

Does Sqoop stdout include the text: **Quest Data Connector for Oracle and Hadoop can be called by Sqoop!**

If it does then Sqoop can load **oraoop-*version*.jar** and execute a method of class **com.quest.oraoop.OraOopManagerFactory.** See "Confirm Quest Data Connector for Oracle and Hadoop Can Accept The Import Job" (page 39) for more information.

Note: The class **com.quest.oraoop.OraOopManagerFactory** is specified in the **$SQOOP_HOME/conf/managers.d/oraoop** file.

Confirm Quest Data Connector for Oracle and Hadoop Can Accept The Import Job

If the output of the Sqoop command from the previous step includes the following text then Quest Data Connector for Oracle and Hadoop has accepted responsibility for the Sqoop import job. Go to the next step: See "Confirm Quest Data Connector for Oracle and Hadoop Can Initialize The Oracle Session" (page 40) for more information.

```
**********************************************************************
*** Using Quest Data Connector for Oracle and Hadoop  1.6.x.xxx ***
***             Copyright 2012 Quest Software, Inc.              ***
***                     ALL RIGHTS RESERVED.                     ***
**********************************************************************
```

If Quest Data Connector for Oracle and Hadoop did not accept the import job

Consult Sqoop stdout (standard output) for the reason.

An example reason may be: `The Oracle object identified by the --table argument is not a table; perhaps it's an Oracle view instead.`

Quote Oracle Owners And Tables

If the owner of the Oracle table needs to be quoted, use:	`$ sqoop import ... --table "\"\"Scott\".customers\""`
	This is the equivalent of:
	`"Scott".customers`

If the Oracle table needs to be quoted, use:	`$ sqoop import ... --table "\"scott.\"Customers\"\""`
	This is the equivalent of:
	`scott."Customers"`

If both the owner of the Oracle table and the table itself needs to be quoted, use:	`$ sqoop import ... --table "\"\"Scott\".\"Customers\"\""`
	This is the equivalent of:
	`"Scott"."Customers"`

Notes:

- The HDFS output directory is called something like: `/user/guy/"Scott"."Customers"`
- If a table name contains a **$** character, it may need to be escaped within your Unix shell. For example, the **dr$object** table in the **ctxsys** schema would be referred to as: **$ sqoop import ... --table "ctxsys.dr\$object"**

Quote Oracle Columns

If a column name of an Oracle table needs to be quoted, use :	`$ sqoop import ... --table customers --columns "\"\"first name\"\""` This is the equivalent of: `select "first name" from customers`

Note: At the time of writing this guide, Sqoop did not support this ability. Refer to the Release Notes for more information.

Confirm Quest Data Connector for Oracle and Hadoop Can Initialize The Oracle Session

If the Sqoop output includes feedback such as the following then the configuration properties contained within **oraoop-site-template.xml** and **oraoop-site.xml** have been loaded by Hadoop and can be accessed by Quest Data Connector for Oracle and Hadoop.

```
10/08/04 13:19:18 INFO oraOop.OracleConnectionFactory: Initializing Oracle
session with SQL : alter session set time_zone = 'US/Hawaii'
```

Check The Sqoop Debug Logs for Error Messages

For more information about any errors encountered during the Sqoop import, refer to the log files generated by each of the (by default 4) mappers that performed the import.

The logs can be obtained via your Map-Reduce Job Tracker's web page.

Include these log files with any requests you make for assistance on the Sqoop User Group web site.

Troubleshooting Export Data

Confirm The Driver Is Operational

Check the Oracle JDBC driver is located in the appropriate locations. See "Install / Upgrade Quest Data Connector for Oracle and Hadoop" (page 11) for more information.

Check Tables Are Compatible

Check tables particularly in the case of a parsing error.

- Ensure the fields contained with the HDFS file and the columns within the Oracle table are identical. If they are not identical, the Java code dynamically generated by Sqoop to parse the HDFS file will throw an error when reading the file – causing the export to fail. When creating a table in Oracle ensure the definitions for the table template are identical to the definitions for the HDFS file.
- Ensure the data types in the table are supported. See "Supported Data Types" (page 14) for more information.
- Are date and time zone based data types used? See "Export Date And Timestamp Data Types into Oracle" (page 28) for more information.

Parallelization

```
-D oraoop.export.oracle.parallelization.enabled=false
```

If you see a parallelization error you may decide to disable parallelization on Oracle queries.

Check oraoop.oracle.append.values.hint.usage

The oraoop.oracle.append.values.hint.usage parameter should not be set to ON if the Oracle table contains either a BINARY_DOUBLE or BINARY_FLOAT column and the HDFS file being exported contains a NULL value in either of these column types. Doing so will result in the error: `ORA-12838: cannot read/modify an object after modifying it in parallel`.

Turn On Verbose

Turn on verbose on the Sqoop command line.

```
--verbose
```

Check Sqoop stdout (standard output) and the mapper logs for information as to where the problem may be.

Appendix: Sqoop Support

If this document is unable to resolve problems you may encounter, please post your questions to the Sqoop user group:

https://groups.google.com/a/cloudera.org/group/sqoop-user

Appendix: Contact Quest

Contact Quest Software

Email:	info@quest.com
Mail:	Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA
Web site:	www.quest.com

Refer to our Web site for regional and international office information.

About Quest Software

Now more than ever, organizations need to work smart and improve efficiency. Quest Software creates and supports smart systems management products—helping our customers solve everyday IT challenges easier and faster. Learn more at www.quest.com.

Appendix: Third Party Contributions

Quest Data Connector for Oracle and Hadoop contains some third party components (listed below). Copies of their licenses may be found at http://www.quest.com/legal/third-party-licenses.aspx.

Component	License or Acknowledgement
JUnit 4.8.1	Common Public License - v 1.0
Log4J 1.2.15	Copyright © 2010 The Apache Software Foundation. Developed by the Apache Software Foundation (http://www.apache.org.) Licensed under the Apache Software License, Version 2.0.