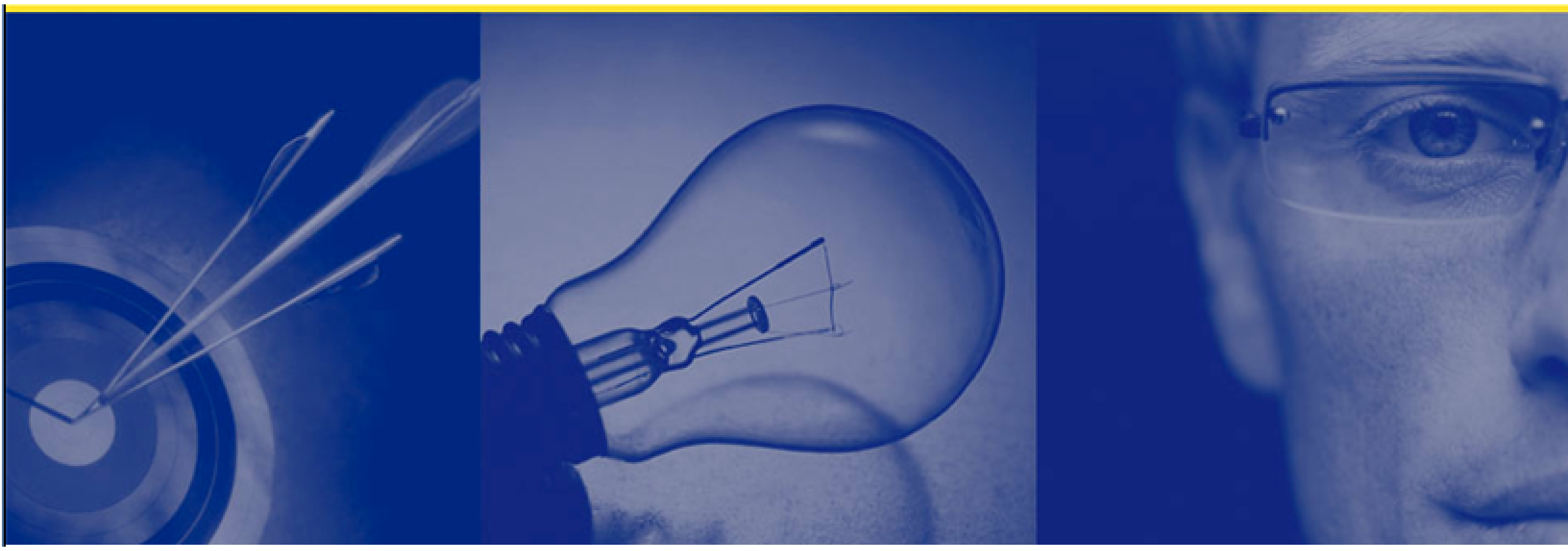
# OraOop 1.0

User Guide



OraOop 1.0
User Guide
October 2010

# Table of Contents

# About OraOop

## What Is OraOop?

OraOop is an optional plugin to Sqoop 1.1.0. It facilitates the movement of data from Oracle into Hadoop. OraOop inspects each Sqoop job and assumes responsibility for the ones it can perform better than the Oracle manager built into Sqoop.

OraOop 1.0 accepts responsibility for Sqoop jobs with the following attributes:

- Oracle-related
- **Import** — Not **export** jobs, **eval** jobs etc.
- Non-Incremental
- Table-Based — Jobs where the table argument is used and the specified object is a table.
- 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

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 OraOop Works

OraOop 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 OraOop 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.

# Download and Install OraOop

## Download OraOop

Download OraOop from the OraOop web site: http://www.quest.com/ora-oop

OraOop is supplied as a compressed tar archive. The archive file name is of the form **oraoop-*version*.tgz** where *version* is the numeric identifier of the OraOop release, such as *1.0.0.106*.

The archive contains the following files:

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

## Install OraOop

| Step | Description |
|---|---|
| Extract the OraOop archive | Extract the OraOop download archive with the command: `tar xzf oraoop-version.tgz` The archive files are extracted into a sub-directory called **oraoop-*version***. **Note:** Ensure the OraOop 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 OraOop installer | Change directory to the sub-directory to which the OraOop files were extracted: **oraoop-*version*** Execute the OraOop Bourne shell script file: **install.sh** No arguments are necessary. This installs OraOop files into Sqoop directories. |

## OraOop Files In Sqoop Directories

OraOop files are installed into the required locations. OraOop can be installed on any Sqoop client by placing the files listed below in the given locations.

| OraOop file | Location |
|---|---|
| oraoop-*version*.jar | `$SQOOP_HOME/lib` **Note:** The Oracle JDBC driver **ojdbc6.jar** should also be in this directory. |
| oraoop-site-template.xml | `$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 directory in your system as indicated in the *Sqoop User Guide*.

**Note:** For Sqoop 1.1.0 you should be able to put this file in **$SQOOP_HOME/lib/** and have the distributed cache take care of copying it to each node in the cluster.

# Oracle and OraOop

## Oracle Roles and Privileges

When performing a Sqoop import with OraOop the user connecting to Oracle requires the following privileges:

- create session

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

- select on dba_tab_columns
- select on dba_objects
- select on dba_extents

## Supported Data Types

The following Oracle column types are supported by OraOop:

| | |
|---|---|
| BLOB | NCLOB |
| CHAR | NUMBER |
| CLOB | NVARCHAR2 |
| DATE | ROWID |
| FLOAT | TIMESTAMP |
| INTERVAL DAY TO SECOND | TIMESATMP WITH TIME ZONE |
| INTERVAL YEAR TO MONTH | TIMESTAMP WITH LOCAL TIME ZONE |
| LONG | VARCHAR2 |
| NCHAR | |

All other Oracle column types are NOT supported. Example Oracle column types NOT supported by OraOop include:

| | | |
|---|---|---|
| All of the ANY types | BFILE | RAW |
| All of the MEDIA types | BINARY_DOUBLE | URITYPE |
| All of the SPATIAL types | BINARY_FLOAT | UROWID |
| Any type referred to as UNDEFINED | LONG RAW | XMLTYPE |
| | MLSLABEL | |

# Execute Sqoop with OraOop

## Execute Sqoop

Ensure OraOop is installed before you execute Sqoop with OraOop. See "Install OraOop" (page 8) for more information.

This is an example Sqoop command to import data from Oracle using OraOop. OraOop requires the connection string starts with **jdbc:oracle**

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

When Sqoop successfully loads OraOop and OraOop accepts the job the following text is output:

```
***********************************
***    Using OraOop 1.0.x.xxx   ***
*** © 2010 Quest Software, Inc. ***
***    ALL RIGHTS RESERVED.     ***
***********************************
```

## Kill OraOop 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 OraOop import job (via killing the sessions in Oracle) you need to prevent Map-Reduce from re-attempting failed jobs. This is done via the Sqoop command-line switch.

```
$ sqoop import -D mapred.map.max.attempts=1 ...
```

Under these conditions, when an OraOop import is started, instructions similar to the following are sent to the console:

```
10/08/13  14:35:24  INFO  oraOop.OraOopManagerFactory:
This  OraOop  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;
```

If this statement is executed on the Oracle database it kills all OraOop sessions associated with this Sqoop job, thereby terminating the Map-Reduce job.

5

# Known Differences Between OraOop and Sqoop Data

This section lists known differences in the data obtained by performing an OraOop import of an Oracle table versus a native Sqoop import of the same table.

## DATE / TIMESTAMP columns do not have a time zone applied to them

Data stored in a DATE or TIMESTAMP column of an Oracle table is not associated with a time zone. Sqoop without OraOop 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 OraOop 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.`

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

**Note:** In order for OraOop 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.

## TIMEZONE-based column types retain time zone information

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 OraOop 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 OraOop 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 OraOop imports the example timestamps as follows. OraOop 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
```

## LOCAL TIMEZONE-based column types explicitly state their time zone

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 OraOop 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 OraOop 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 OraOop explicitly states the (local) time zone portion of the data imported into Hadoop. The local time zone is as per the system executing the Sqoop job. OraOop would import these two timestamps as:

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

# Configure OraOop

## Begin with oraoop-site-template.xml

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

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

1. Find **oraoop-site-template.xml** in the Sqoop configuration directory. See "OraOop Files In Sqoop Directories" (page 8) 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 OraOop.

The default statements include:

| |
|---|
| **alter session set time_zone = '{oracle.sessionTimeZone\|GMT}';**<br>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.<br>**Notes:**<br>• There is an explanation to the text within the curly-braces. See "Expressions In oraoop-site.xml" (page 20) for more information..<br>• 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;**<br>This statement instructs Oracle to not parallelize SQL statements executed by the OraOop sessions. This Oracle feature is disabled, because the Map/Reduce job launched by Sqoop is the mechanism being used to parallelize the import of the data. |
| **alter session set "_serial_direct_read"=true;**<br>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 OraOop, therefore diminishing its capacity to cache higher prioritized data. Hence, this statement is intended to minimize OraOop'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 statement 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.<br><br>A Sqoop command like:<br>`sqoop import -D oraoop.table.import.where.clause.location=SUBSPLIT --table JUNK --where “owner like 'G%'”`<br><br>Generates SQL query of the form:<br>`SELECT`<br>`OWNER,OBJECT_NAME`<br>`FROM JUNK`<br>`  WHERE ((rowid >=`<br>`  dbms_rowid.rowid_create(1, 113320, 1024, 4223664, 0)`<br>`  AND`<br>`  rowid <=`<br>`  dbms_rowid.rowid_create(1, 113320, 1024, 4223671, 32767)))`<br>` AND (owner like 'G%')`<br>`UNION ALL`<br>`SELECT OWNER,OBJECT_NAME`<br>` FROM JUNK`<br>` WHERE ((rowid >=`<br>`  dbms_rowid.rowid_create(1, 113320, 1024, 4223672, 0)`<br>`  AND rowid <=`<br>`  dbms_rowid.rowid_create(1, 113320, 1024, 4223679, 32767)))`<br>` 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.<br><br>A Sqoop command like:<br>`sqoop import -D oraoop.table.import.where.clause.location=SPLIT --table JUNK --where “rownum <= 10”`<br><br>Generates SQL query of the form:<br>`SELECT`<br>` OWNER,OBJECT_NAME`<br>` FROM (` |

| Value | Description |
|---|---|
| | `  SELECT`<br>`   OWNER,OBJECT_NAME`<br>`   FROM JUNK`<br>`   WHERE ((rowid >=`<br>`    dbms_rowid.rowid_create(1, 113320, 1024, 4223664, 0)`<br>`    AND`<br>`    rowid <=`<br>`    dbms_rowid.rowid_create(1, 113320, 1024, 4223671,`<br>`32767)))`<br>`  UNION ALL`<br>`  SELECT OWNER,OBJECT_NAME`<br>`   FROM JUNK`<br>`   WHERE ((rowid >=`<br>`    dbms_rowid.rowid_create(1, 113320, 1024, 4223672, 0)`<br>`    AND rowid <=`<br>`    dbms_rowid.rowid_create(1, 113320, 1024, 4223679,`<br>`32767)))`<br>`)`<br>`WHERE` **`rownum <= 10`**<br><br>**Note:** In this example, there are up to 10 rows imported per 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.

## mapred.map.tasks.speculative.execution

By default, speculative execution is disabled for OraOop imports. This avoids placing redundant load on the Oracle database.

For example, having multiple Map-Reduce mappers querying the same data from the Oracle database to see which is going to finish first is disabled.

## oraoop.block.allocation

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

| 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.<br>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.<br>Use of this method is generally not recommended. |

## oraoop.locations

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...
```

## 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 set, then this statement would use the specified default value and would become: | `alter session set time_zone =` **`'GMT'`** |

**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.

# Known Issues with OraOop

| Feature | Known Issue | Defect ID |
|---|---|---|
| General | Column names that are explicitly listed for a Sqoop import, which are escaped cause an NullPointerException in Sqoop<br>`$ sqoop import ... --table customers --columns "\"\"first name\"\""` | This has been raised as Jira: SQOOP-92 |
| | Null values from Oracle are imported into Hive (via Sqoop) as the string "null". This is due to Sqoop serializing the value NULL as a string containing the four letters "null".<br>Executing the following query always returns zero, even with null values in Oracle:<br>`hive> select count(1) from customers where surname is null;`<br>To avoid this issue, query for:<br>`hive> select count(1) from customers where surname = 'null';` | N/A |
| | NCLOB and NCHAR data that is UTF-16 encoded is converted to UTF-8 encoding during a Sqoop import. (You may or may not consider this a problem.) | N/A |
| | OraOop fails to properly import an Index-Organized table - possibly claiming that it contains no data.<br>To avoid this issue it is recommended that a Sqoop import of an index-organized table be performed with OraOop disabled. See "Disable OraOop" (page 22) for more information. | N/A |

# Troubleshooting OraOop

## Disable OraOop

To disable OraOop add the following argument to the Sqoop command line: `-D oraoop.disabled=true`

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

OraOop does not accept Oracle related jobs while it is disabled.

Verify OraOop is disabled by checking Sqoop stdout (standard output). The stdout should have no reference to OraOop or a reference to indicate OraOop was disabled.

**Note:** Disabling OraOop via the Sqoop command-line disables OraOop for the Sqoop job. It does not permanently switch off OraOop.

## Confirm Sqoop Can Import Oracle Without OraOop

Disable OraOop 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 OraOop is disabled**
See "Confirm the Imported CSV Data Can Be Correctly Parsed" (page 23) for more information.

**If Sqoop CANNOT import the Oracle data while OraOop is disabled**

1. Check the Oracle JDBC driver is located in the appropriate locations. See "Install OraOop" (page 8) 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: OraOop 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 OraOop. Turn on verbose.

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

Does Sqoop stdout include the text: **OraOop 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 OraOop Can Accept the Import job" (page 24) for more information.

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

## Confirm OraOop Can Accept the Import job

If the output of the Sqoop command from the previous step includes the following text then OraOop has accepted responsibility for the Sqoop import job. Go to the next step: See "Confirm OraOop Can Initialize the Oracle Session" (page 25) for more information.

```
***********************************
***    Using OraOop 1.0.x.xxx    ***
*** © 2010 Quest Software, Inc. ***
***    ALL RIGHTS RESERVED.      ***
***********************************
```

**If OraOop 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. See "Known Issues with OraOop" (page 21) for more information.

## Confirm OraOop 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 OraOop.

```
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.

## 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.<br>World Headquarters<br>5 Polaris Way<br>Aliso Viejo, CA 92656<br>USA |
| Web site | www.quest.com |

See 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.