

Oracle® Big Data SQL

User's Guide

Release 3 (3.0)

E71333-05

May 2016

Describes the Oracle Big Data SQL software available to administrators and software developers.

Copyright © 2012, 2016, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	vii
Audience	vii
Related Documents.....	vii
Conventions.....	vii
Backus-Naur Form Syntax.....	viii
 Changes in This Release for Oracle Big Data SQL	 ix
 1 Introducing Oracle Big Data SQL	
1.1 What Is Oracle Big Data SQL?	1-1
1.1.1 About Oracle External Tables	1-1
1.1.2 About Predicate Push Down	1-1
1.1.3 About the Access Drivers for Oracle Big Data SQL	1-3
1.1.4 About Smart Scan Technology	1-3
1.1.5 About Storage Indexes.....	1-4
 2 Installing Oracle Big Data SQL	
2.1 Oracle Big Data SQL Compatibility Matrix	2-1
2.2 Installing On Oracle Big Data Appliance and the Oracle Exadata Database Machine	2-1
2.2.1 Performing the Installation	2-1
2.2.2 Running the Post-Installation Script for Oracle Big Data SQL	2-2
2.2.3 About Data Security with Oracle Big Data SQL	2-8
2.2.4 Enabling Oracle Big Data SQL Access to a Kerberized Cluster.....	2-8
2.2.5 Starting and Stopping the Big Data SQL Agent.....	2-9
2.3 Installing Oracle Big Data SQL on Other Hadoop Systems	2-10
2.3.1 Downloading Oracle Big Data SQL	2-10
2.3.2 Prerequisites for Installing on an HDP Cluster	2-11
2.3.3 Prerequisites for Installing on a CDH Cluster	2-12
2.3.4 Installation Overview	2-13
2.3.5 Installing on the Hadoop Cluster Management Server	2-14
2.3.6 Creating the Database-Side Installation Bundle	2-16

2.3.7	Installing on the Oracle Database Server	2-19
2.3.8	Uninstalling Oracle Big Data SQL	2-21
2.3.9	Securing Big Data SQL	2-22

3 Using Oracle Big Data SQL for Data Access

3.1	Creating an Oracle External Table for Hive Data	3-1
3.1.1	Obtaining Information About a Hive Table	3-1
3.1.2	Using the CREATE_EXTDDL_FOR_HIVE Function	3-2
3.1.3	Developing a CREATE TABLE Statement for ORACLE_HIVE	3-3
3.2	Creating an Oracle External Table for Oracle NoSQL Database	3-4
3.2.1	Creating a Hive External Table for Oracle NoSQL Database	3-5
3.2.2	Creating the Oracle Database Table for Oracle NoSQL Data	3-6
3.2.3	About Column Data Type Mappings.....	3-6
3.2.4	Example of Accessing Data in Oracle NoSQL Database	3-7
3.3	Creating an Oracle External Table for Apache HBase	3-10
3.3.1	Creating a Hive External Table for HBase.....	3-10
3.3.2	Creating the Oracle Database Table for HBase	3-10
3.4	Creating an Oracle External Table for HDFS Files	3-11
3.4.1	Using the Default Access Parameters with ORACLE_HDFS	3-11
3.4.2	Overriding the Default ORACLE_HDFS Settings.....	3-11
3.5	About the SQL CREATE TABLE Statement	3-13
3.5.1	Basic Syntax.....	3-13
3.5.2	About the External Table Clause.....	3-13
3.6	About Data Type Conversions.....	3-15
3.7	Querying External Tables	3-16
3.7.1	Granting User Access.....	3-16
3.7.2	About Error Handling	3-17
3.7.3	About the Log Files	3-17
3.8	About Oracle Big Data SQL on the Database Server (Oracle Exadata Machine or Other) .	3-17
3.8.1	About the Common Directory.....	3-17
3.8.2	Common Configuration Properties	3-17
3.8.3	About the Cluster Directory	3-19
3.8.4	About Permissions	3-20

4 Copying Oracle Tables to Hadoop

4.1	What Is Copy to Hadoop?	4-1
4.2	Getting Started Using Copy to Hadoop	4-1
4.3	Installing Copy to Hadoop	4-2
4.3.1	Prerequisites for Copy to Hadoop	4-2
4.3.2	Installing Copy to Hadoop in an Oracle Big Data Appliance/Oracle Exadata Database Machine Environment	4-2
4.3.3	Installing Copy to Hadoop on Other Systems	4-2
4.4	Generating the Data Pump Files.....	4-3

4.4.1	About Data Pump Format Files.....	4-3
4.4.2	Identifying the Target Directory	4-3
4.4.3	About the CREATE TABLE Syntax	4-4
4.4.4	Copying the Files to HDFS.....	4-4
4.5	Creating a Hive Table.....	4-5
4.5.1	About Hive External Tables.....	4-5
4.5.2	About Column Mappings	4-5
4.5.3	About Data Type Conversions	4-5
4.6	Example Using the Sample Schemas	4-6
4.6.1	About the Sample Data.....	4-6
4.6.2	Creating the EXPDIR Database Directory	4-7
4.6.3	Creating Data Pump Format Files for Customer Data	4-7
4.6.4	Verifying the Contents of the Data Files	4-8
4.6.5	Copying the Files into Hadoop	4-9
4.6.6	Creating a Hive External Table	4-9
4.6.7	Querying the Data in Hive.....	4-9

5 Oracle Big Data SQL Reference

5.1.1	DBMS_HADOOP PL/SQL Package	5-1
5.1.1.1	CREATE_EXTDDL_FOR_HIVE.....	5-1
5.1.2	CREATE TABLE ACCESS PARAMETERS Clause.....	5-3
5.1.2.1	Syntax Rules for Specifying Properties	5-3
5.1.2.2	ORACLE_HDFS Access Parameters.....	5-4
5.1.2.3	ORACLE_HIVE Access Parameters	5-5
5.1.2.4	com.oracle.bigdata.colmap	5-6
5.1.2.5	com.oracle.bigdata.datamode.....	5-7
5.1.2.6	com.oracle.bigdata.erroropt.....	5-8
5.1.2.7	com.oracle.bigdata.fields.....	5-9
5.1.2.8	com.oracle.bigdata.fileformat.....	5-10
5.1.2.9	com.oracle.bigdata.log.exec	5-11
5.1.2.10	com.oracle.bigdata.log.qc.....	5-12
5.1.2.11	com.oracle.bigdata.overflow	5-13
5.1.2.12	com.oracle.bigdata.rowformat	5-14
5.1.2.13	com.oracle.bigdata.tablename	5-15
5.1.3	Static Data Dictionary Views for Hive.....	5-16
5.1.3.1	ALL_HIVE_DATABASES.....	5-16
5.1.3.2	ALL_HIVE_TABLES.....	5-17
5.1.3.3	ALL_HIVE_COLUMNS.....	5-18
5.1.3.4	DBA_HIVE_DATABASES	5-19
5.1.3.5	DBA_HIVE_TABLES	5-19
5.1.3.6	DBA_HIVE_COLUMNS	5-19
5.1.3.7	USER_HIVE_DATABASES.....	5-19
5.1.3.8	USER_HIVE_TABLES.....	5-20

5.1.3.9 USER_HIVE_COLUMNS.....	5-20
--------------------------------	------

Appendices

A Licensing Information

A.1 Oracle Big Data SQL	A-1
-------------------------------	-----

Index

Preface

The *Oracle Big Data SQL User's Guide* describes how to install and manage the Oracle Big Data SQL product.

Audience

This guide is intended for installers and users of Oracle Big Data SQL, including:

- Application developers
- Data analysts
- Data scientists
- Database administrators
- System administrators

The guide assumes that the reader has sufficient background knowledge about the database server and the particular Hadoop platform which will host the software in order to follow the instructions successfully.

Related Documents

Users installing Oracle Big Data SQL on the Oracle Big Data Appliance can get more information about Oracle's Big Data solutions and the Oracle Big Data Appliance in particular at the [Oracle Help Center](#)

The following publications are recommended:

- *Oracle Big Data Appliance Owner's Guide*
- *Oracle Big Data Appliance Software User's Guide*
- *Oracle Big Data Connectors User's Guide*

Users installing Oracle Big Data SQL on Hortonworks HDP should refer to the Hortonworks documentation site at <http://docs.hortonworks.com/index.html> for supplementary information

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
# prompt	The pound (#) prompt indicates a command that is run as the Linux <code>root</code> user.

Backus-Naur Form Syntax

The syntax in this reference is presented in a simple variation of Backus-Naur Form (BNF) that uses the following symbols and conventions:

Symbol or Convention	Description
[]	Brackets enclose optional items.
{ }	Braces enclose a choice of items, only one of which is required.
	A vertical bar separates alternatives within brackets or braces.
...	Ellipses indicate that the preceding syntactic element can be repeated.
delimiters	Delimiters other than brackets, braces, and vertical bars must be entered as shown.
boldface	Words appearing in boldface are keywords. They must be typed as shown. (Keywords are case-sensitive in some, but not all, operating systems.) Words that are not in boldface are placeholders for which you must substitute a name or value.

Changes in This Release for Oracle Big Data SQL

A significant change in Oracle Big Data SQL 3.0 is the expansion of support to non-Oracle servers in addition to support for Oracle Big Data Appliance/Oracle Exadata Database Machine installations. Another new feature to note is predicate push down.

Predicate Push Down

Big Data SQL 3.0 takes advantage of predicate off-loading support by pushing SARGable (*Search ARGument ABLE*) predicates from Oracle Database into supporting systems. See [About Predicate Push Down](#)

Support for Servers Other than Oracle Engineered Systems

Prior to Release 3.0, Oracle Big Data SQL supported only Oracle Engineered Systems. Specifically, Oracle Big Data Appliance (on the Hadoop side), and Oracle Exadata Database Machine (on the database server side). As of Release 3.0, Oracle Big Data SQL also runs on the **Hortonworks Hadoop Data Platform** and on **Cloudera's Distribution Including Apache Hadoop (CDH)** systems other than Oracle Big Data Appliance.

Big Data SQL also now supports Oracle Database running on servers other than Oracle Exadata Database Machine.

See the *Oracle Big Data SQL Master Compatibility Matrix* (Doc ID 2119369.1) in [My Oracle Support](#) for supported operating systems, Hadoop distributions, and Oracle database configurations.

Note: Oracle Big Data SQL does not yet support connections between Oracle Engineered Systems and other servers. The following configurations are not supported in Release 3.0.

- Oracle Exadata Database Machines connected to a Hadoop system other than Oracle Big Data Appliance.
- Non-Exadata Oracle Database servers connected to Oracle Big Data Appliance.

The Oracle Database server used in conjunction with either the HDP or CDH Hadoop systems described above must be a non-Exadata machine.

Introducing Oracle Big Data SQL

1.1 What Is Oracle Big Data SQL?

Oracle Big Data SQL supports queries against vast data sets stored in multiple big data sources, including Apache Hive, HDFS, Oracle NoSQL Database, and Apache HBase. It enables unified query for distributed data and therefore the ability to view and analyze data from disparate data stores seamlessly, as if it were all stored in an Oracle database.

Oracle Big Data SQL supports the complete Oracle SQL syntax. You can execute highly complex SQL `SELECT` statements against data in the Hadoop ecosystem, either manually or through your existing applications. For example, if you are a user of Oracle Advanced Analytics, Oracle Big Data SQL enables you to extend your Oracle Database data mining models to big data in Hadoop.

The following sections provide further details:

- [About Oracle External Tables](#)
- [About Predicate Push Down](#)
- [About the Access Drivers for Oracle Big Data SQL](#)
- [About Smart Scan Technology](#)
- [About Storage Indexes](#)

1.1.1 About Oracle External Tables

Oracle Big Data SQL provides external tables with next generation performance gains. An **external table** is an Oracle Database object that identifies and describes the location of data outside of a database. You can query an external table using the same SQL `SELECT` syntax that you use for any other database tables.

External tables use **access drivers** to parse the data outside the database. Each type of external data requires a unique access driver. Oracle Big Data SQL includes two access drivers for big data: one for data that has metadata defined in Apache Hive, and the other for accessing data stored in the Hadoop Distributed File System, with metadata specified only by an Oracle administrator.

1.1.2 About Predicate Push Down

Many Big Data systems support some level of predicate off-loading, either through the filetype itself (e.g. Apache Parquet), or through Hive's partitioning and StorageHandler APIs. Big Data SQL takes advantage of these off-load capabilities by pushing SARGable (*Search ARGument ABLE*) predicates from the Oracle Database into

supporting systems. For example, predicate push down enables the following automatic behaviors:

- Queries against partitioned Hive tables are pruned, based on filter predicates on partition columns.
- Queries against Apache Parquet and Apache ORC files reduce I/O by utilizing SARGable predicates when reading the files from disk.
- Queries against Oracle NoSQL Database or Apache HBase use SARGable predicates to drive subscans of data in the remote data store.

Required Datatypes to Enable Predicate Push Down

Predicate push down requires that certain mappings between Hive Datatypes and Oracle Datatypes be present. These mappings are described in the following table.

Datatype	Mapped To
Hive Datatype	Must be mapped to Oracle Datatype(s)
<i>Character Datatypes</i>	
CHAR(m)	CHAR(n), VARCHAR2(n) where n is \geq m
VARCHAR(m)	CHAR(n), VARCHAR2(n) where n is \geq m
string	CHAR(n), VARCHAR2(n)
<i>Datetime Datatypes</i>	
DATE	DATE
TIMESTAMP	TIMESTAMP(9) Hive TIMESTAMP has nanoseconds, 9 digit fractional seconds
<i>Simple Numeric Datatypes</i>	
TINYINT	NUMBER(3) preferably, but NUMBER or NUMBER(n) for any value of n is OK
SMALLINT	NUMBER(5) preferably, but NUMBER or NUMBER(n) for any value of n is OK
INT	NUMBER(10) preferably, but NUMBER or NUMBER(n) for any value of n is OK
BIGINT	NUMBER(19) preferably, but NUMBER or NUMBER(n) for any value of n is OK
DECIMAL(m)	NUMBER(n) where m = n preferably, but NUMBER or NUMBER(n) for any value of n is OK
<i>IEEE754 Binary Floating Point Datatypes</i>	
FLOAT	BINARY_FLOAT
DOUBLE	BINARY_DOUBLE

Miscellaneous Datatypes

BINARY	RAW(n)
BOOLEAN	CHAR(n), VARCHAR2(n) where n is ≥ 5 , values 'TRUE', 'FALSE'
BOOLEAN	NUMBER(1) preferably, but NUMBER or NUMBER(n) for any value of n is OK values 0 (false), 1 (true)

1.1.3 About the Access Drivers for Oracle Big Data SQL

By querying external tables, you can access data stored in HDFS and Hive tables as if that data was stored in tables in an Oracle database. Oracle Database accesses the data by using the metadata provided when the external table was created.

Oracle Database 12.1.0.2 supports two new access drivers for Oracle Big Data SQL:

- **ORACLE_HIVE:** Enables you to create Oracle external tables over Apache Hive data sources. Use this access driver when you already have Hive tables defined for your HDFS data sources. **ORACLE_HIVE** can also access data stored in other locations, such as HBase, that have Hive tables defined for them.
- **ORACLE_HDFS:** Enables you to create Oracle external tables directly over files stored in HDFS. This access driver uses Hive syntax to describe a data source, assigning default column names of COL_1, COL_2, and so forth. You do not need to create a Hive table manually as a separate step.

Instead of acquiring the metadata from a Hive metadata store the way that **ORACLE_HIVE** does, the **ORACLE_HDFS** access driver acquires all of the necessary information from the access parameters. The **ORACLE_HDFS** access parameters are required to specify the metadata, and are stored as part of the external table definition in Oracle Database.

Oracle Big Data SQL uses these access drivers to optimize query performance.

1.1.4 About Smart Scan Technology

External tables do not have traditional indexes, so that queries against them typically require a full table scan. However, Oracle Big Data SQL extends SmartScan capabilities (such as filter-predicate off-loads) to Oracle external tables with the installation of the Big Data SQL processing agent on the DataNodes of the Hadoop cluster. This technology enables the Hadoop cluster to discard a huge portion of irrelevant data—up to 99 percent of the total—and return much smaller result sets to the Oracle Database server. End users obtain the results of their queries significantly faster, as the direct result of a reduced load on Oracle Database and reduced traffic on the network.

See Also:

Oracle Database Concepts for a general introduction to external tables and pointers to more detailed information in the Oracle Database documentation library

1.1.5 About Storage Indexes

Oracle Big Data SQL maintains Storage Indexes automatically, which is transparent to Oracle Database. Storage Indexes contain the summary of data distribution on a hard disk for the data that is stored in HDFS. Storage Indexes reduce the I/O operations cost and the CPU cost of converting data from flat files to Oracle Database blocks.

Storage Indexes can be used only for the external tables that are based on HDFS and are created using either the ORACLE_HDFS driver or the ORACLE_HIVE driver. Storage Indexes cannot be used for the external tables that use StorageHandlers, such as Apache HBase and Oracle NoSQL.

A Storage Index is a collection of in-memory region indexes, and each region index stores summaries for up to 32 columns. There is one region index for each split. The content stored in one region index is independent of the other region indexes. This makes them highly scalable, and avoids latch contention.

Storage Indexes maintain the minimum and maximum values of the columns of a region for each region index. The minimum and maximum values are used to eliminate unnecessary I/O, also known as I/O filtering. The cell XT granule I/O bytes saved by the Storage Indexes statistic, available in the V\$SYSSTAT view, shows the number of bytes of I/O saved using Storage Indexes.

See Also:

Oracle® Database Reference for information about V\$SYSSTAT view

Queries using the following comparisons are improved by the Storage Indexes:

- Equality (=)
- Inequality (<, !=, or >)
- Less than or equal (<=)
- Greater than or equal (>=)
- IS NULL
- IS NOT NULL

Storage Indexes are built automatically after Oracle Big Data SQL service receives a query with a comparison predicate that is greater than the maximum or less than the minimum value for the column in a region.

Note:

- The effectiveness of Storage Indexes can be improved by ordering the rows in a table based on the columns that frequently appear in the WHERE query clause.
 - Storage Indexes work with any non-linguistic data type, and works with linguistic data types similar to non-linguistic index.
-
-

Example 1-1 Elimination of Disk I/O with Storage Indexes

The following figure shows a table and region indexes. The values in the table range from 1 to 8. One region index stores the minimum 1, and the maximum of 5. The other region index stores the minimum of 3, and the maximum of 8.

Table				Index	
A	B	C	D		
	1			}	Min B = 1 Max B = 5
	3				
	5				
	5			}	Min B = 3 Max B = 8
	8				
	3				

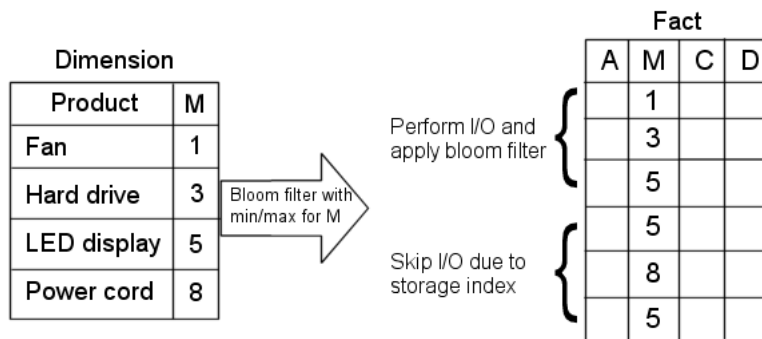
I/O eliminated by using storage index

For a query such as `SELECT * FROM TABLE WHERE B < 2`, only the first set of rows match. Disk I/O is eliminated because the minimum and maximum of the second set of rows do not match the WHERE clause of the query.

Example 1-2 Improved Join Performance Using Storage Indexes

Using Storage Indexes allows table joins to skip unnecessary I/O operations. For example, the following query would perform an I/O operation and apply a Bloom filter to only the first block of the fact table.

```
SELECT count(*) from fact, dim where fact.m=dim.m and
dim.name="Hard drive"
```



The I/O for the second block of the fact table is completely eliminated by Storage Indexes as its minimum/maximum range (5,8) is not present in the Bloom filter.

Installing Oracle Big Data SQL

Oracle Big Data SQL 3.0 can connect Oracle Database to the Hadoop environment on Oracle Big Data Appliance, other systems based on CDH (Cloudera's Distribution including Apache Hadoop), HDP (Hortonworks Data Platform), and potentially other non-CDH Hadoop systems.

The procedures for installing Oracle Big Data SQL in these environments differ. To install the product in your particular environment, see the appropriate section:

- [Installing On Oracle Big Data Appliance and the Oracle Exadata Database Machine](#)
See this section for installation on Oracle Big Data Appliance and Exadata servers only.
- [Installing Oracle Big Data SQL on Other Hadoop Systems](#)
See this section for installation on both CDH (excluding Oracle Big Data Appliance) and non-CDH (specifically, HDP) systems.

2.1 Oracle Big Data SQL Compatibility Matrix

See the *Oracle Big Data SQL Master Compatibility Matrix* (Doc ID 2119369.1) in [My Oracle Support](#) for up-to-date information on Big Data SQL compatibility with the following:

- Oracle Engineered Systems.
- Other systems.
- Linux OS distributions and versions.
- Hadoop distributions.
- Oracle Database releases, including required patches.

2.2 Installing On Oracle Big Data Appliance and the Oracle Exadata Database Machine

To use Oracle Big Data SQL on an Oracle Exadata Database Machine connected to Oracle Big Data Appliance, you must install the Oracle Big Data SQL software on both systems.

2.2.1 Performing the Installation

Follow these steps to install the Oracle Big Data SQL software on Oracle Big Data Appliance and Oracle Exadata Database Machine.

Note:

This procedure is not applicable to the installation of Oracle Big Data SQL on systems other than Oracle Big Data Appliance and Oracle Exadata Database Machine.

The January 2016 Bundle Patch (12.1.0.2.160119 BP) for Oracle Database must be pre-installed on the Exadata Database Machine. Earlier Bundle Patches are not supported at this time.

1. Download the Oracle Database one-off patch 22778199.
2. On all Oracle Exadata Database Machine compute servers, install the patch on:

- Grid Infrastructure home
- Oracle Database home

Remember to run the Datapatch part of the Bundle Patch. See the patch README for step-by-step instructions for installing the patch.

3. On Oracle Big Data Appliance, install or upgrade the software to the latest version. See *Oracle Big Data Appliance Owner's Guide*. **However, skip the Oracle Big Data SQL 2.0 installation option in Mammoth and do not enable Oracle Big Data SQL Release 2.0**. If Release 2.0 is present, then it must be uninstalled before installing Oracle Big Data SQL 3.0.

4. On Oracle Big Data Appliance, download the Oracle Big Data SQL 3.0 patch.

```
Patch 22911748: PATCH FOR BIG DATA SQL V3.0.0 ON BDA V4.4.0
FOR ORACLE LINUX 6
```

Follow the instructions in the patch README file.

5. On each Oracle Exadata Database Machine, run the post-installation script.

See [“Running the Post-Installation Script for Oracle Big Data SQL”](#).

You must run the post-installation script on every node of the Exadata database cluster.

You can use Cloudera Manager to verify that Oracle Big Data SQL is up and running.

When you are done, if the cluster is secured by Kerberos then there are additional steps you must perform on both the cluster nodes and on the Oracle Exadata Database Machine. See [Enabling Oracle Big Data SQL Access to a Kerberized Cluster](#).

In the case of an Oracle Big Data Appliance upgrade, the customer is responsible for upgrading the Oracle Database to a supported level before re-running the post-installation script.

2.2.2 Running the Post-Installation Script for Oracle Big Data SQL

Important: Run `bds-exa-install.sh` on every node of the Exadata cluster. If this is not done, you will see RPC connection errors when the BDS service is started.

To run the Oracle Big Data SQL post-installation script:

1. Copy the `bds-exa-install.sh` installation script from the Oracle Big Data Appliance to a temporary directory on the Oracle Exadata Database machine. (Find the script on the node where Mammoth is installed, typically the first node in the cluster.) For example:

```
# curl -O http://bda1node07/bda/bds-exa-install.sh
```

2. Verify the name of the Oracle installation owner and set the executable bit for this user. Typically, the `oracle` user owns the installation. For example:

```
$ ls -l bds-exa-install.sh
$ chown oracle:oinstall bds-exa-install.sh
$ chmod +x bds-exa-install.sh
```

3. Set the following environment variables:

```
$ORACLE_HOME to <database home>
$ORACLE_SID to <correct db SID>
$GI_HOME to <correct grid home>
```

Note:

You can set the grid home with the install script as mentioned in step 5 d instead of setting the `$GI_HOME` as mentioned in this step.

4. Check that `TNS_ADMIN` is pointing to the directory where the right `listener.ora` is running. If the listener is in the default `TNS_ADMIN` location, `$ORACLE_HOME/network/admin`, then there is no need to define the `TNS_ADMIN`. But if the listener is in a non-default location, `TNS_ADMIN` must correctly point to it, using the command:

```
export TNS_ADMIN=<path to listener.ora>
```

5. Perform this step only if the `ORACLE_SID` is in uppercase, else you can proceed to the next step. This is because the install script derives the CRS database resource from `ORACLE_SID`, only if it is in lowercase. Perform the following sequence of steps to manually pass the SID to the script, if it is in uppercase:

- a. Run the following command to list all the resources.

```
$ crsctl stat res -t
```

- b. From the output note down the `ora.<dbresource>.db` resource name.

- c. Run the following command to verify whether the correct `ora.<dbresource>.db` resource name is returned or not.

```
$ ./crsctl stat res ora.<dbresource>.db
```

The output displays the resource names as follows:

```
NAME=ora.<dbresource>.db
TYPE=ora.database.type
TARGET=ONLINE , ONLINE
STATE=ONLINE on <name01>, ONLINE on <name02>
```

- d. Specify the `--db-name=<dbresource>` as additional argument to the install script as follows:

```
./bds-exa-install.sh --db-name=<dbresource>
```

Additionally, you can set the grid home instead of setting the \$GI_HOME as mentioned in step 3, along with the above command as follows:

```
./bds-exa-install.sh --db-name=<dbresource> --grid-home=<grid home>
```

Note:

You can skip the next step, if you performed this step.

6. Run the script as any user who has dba privileges (who can connect to sys as sysdba).

```
./bds-exa-install.sh
```

You must run the script as root in another session when prompted by the script to proceed as the oracle user. For example,

```
$ ./bda-exa-install.sh:
bds-exa-install: root shell script      : /u01/app/oracle/product/12.1.0.2/
dbhome_1/install/bds-root-<cluster-name>-setup.sh
please run as root:
/u01/app/oracle/product/12.1.0.2/dbhome_1/install/bds-root-<rack-name>-clu-
setup.sh
```

A sample output is shown here:

```
ds-exa-install: platform is Linux
bds-exa-install: setup script started at   : Sun Feb 14 20:06:17 PST 2016
bds-exa-install: bds version               : bds-3.0-0.el6.x86_64
bds-exa-install: bda cluster name          : mycluster1
bds-exa-install: bda web server            : mycluster1bda16.us.oracle.com
bds-exa-install: cloudera manager url      : mycluster1bda18.us.oracle.com:7180
bds-exa-install: hive version              : hive-1.1.0-cdh5.5.1
bds-exa-install: hadoop version            : hadoop-2.6.0-cdh5.5.1
bds-exa-install: bds install date          : 02/14/2016 12:00 PST
bds-exa-install: bd_cell version           :
bd_cell-12.1.2.0.100_LINUX.X64_160131-1.x86_64
bds-exa-install: action                   : setup
bds-exa-install: crs                     : true
bds-exa-install: db resource              : orcl
bds-exa-install: database type            : SINGLE
bds-exa-install: cardinality              : 1
bds-exa-install: root shell script        : /u03/app/oracle/product/12.1.0/
dbhome_1/install/bds-root-mycluster1-setup.sh
please run as root:
```

```
/u03/app/oracle/product/12.1.0/dbhome_1/install/bds-root-mycluster1-setup.sh
```

```
waiting for root script to complete, press <enter> to continue checking..
q<enter> to quit
bds-exa-install: root script seem to have succeeded, continuing with setup bds
bds-exa-install: working directory        : /u03/app/oracle/product/12.1.0/
dbhome_1/install
bds-exa-install: downloading JDK
bds-exa-install: working directory        : /u03/app/oracle/product/12.1.0/
dbhome_1/install
bds-exa-install: installing JDK tarball
bds-exa-install: working directory        : /u03/app/oracle/product/12.1.0/
```

```
dbhome_1/bigdatasql/jdk1.8.0_66/jre/lib/security
bds-exa-install: Copying JCE policy jars
/bin/mkdir: cannot create directory `bigdata_config/mycluster1': File exists
bds-exa-install: working directory      : /u03/app/oracle/product/12.1.0/
dbhome_1/bigdatasql/jlib
bds-exa-install: removing old oracle bds jars if any
bds-exa-install: downloading oracle bds jars
bds-exa-install: installing oracle bds jars
bds-exa-install: working directory      : /u03/app/oracle/product/12.1.0/
dbhome_1/bigdatasql
bds-exa-install: downloading           : hadoop-2.6.0-cdh5.5.1.tar.gz
bds-exa-install: downloading           : hive-1.1.0-cdh5.5.1.tar.gz
bds-exa-install: unpacking             : hadoop-2.6.0-cdh5.5.1.tar.gz
bds-exa-install: unpacking             : hive-1.1.0-cdh5.5.1.tar.gz
bds-exa-install: working directory      : /u03/app/oracle/product/12.1.0/
dbhome_1/bigdatasql/hadoop-2.6.0-cdh5.5.1/lib
bds-exa-install: downloading           : cdh-ol6-native.tar.gz
bds-exa-install: creating /u03/app/oracle/product/12.1.0/dbhome_1/bigdatasql/
hadoop_mycluster1.env for hdfs/mapred client access
bds-exa-install: working directory      : /u03/app/oracle/product/12.1.0/
dbhome_1/bigdatasql
bds-exa-install: creating bds property files
bds-exa-install: working directory      : /u03/app/oracle/product/12.1.0/
dbhome_1/bigdatasql/bigdata_config
bds-exa-install: created bigdata.properties
bds-exa-install: created  bigdata-log4j.properties
bds-exa-install: creating default and cluster directories needed by big data
external tables
bds-exa-install: note this will grant default and cluster directories to public!
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_catcon_29579.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon*.log files
for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon_*.lst
files for spool files, if any
catcon.pl: completed successfully
bds-exa-install: granted default and cluster directories to public!
bds-exa-install: mta set to use listener end point : EXTPROC1521
bds-exa-install: mta will be setup
bds-exa-install: creating /u03/app/oracle/product/12.1.0/dbhome_1/hs/admin/
initbds_orcl_mycluster1.ora
bds-exa-install: mta setting agent home as : /u03/app/oracle/product/12.1.0/
dbhome_1/hs/admin
bds-exa-install: mta shutdown           : bds_orcl_mycluster1
bds-exa-install: registering crs resource : bds_orcl_mycluster1
bds-exa-install: using dependency db resource of orcl
bds-exa-install: starting crs resource   : bds_orcl_mycluster1
CRS-2672: Attempting to start 'bds_orcl_mycluster1' on 'mycluster1bda09'
CRS-2676: Start of 'bds_orcl_mycluster1' on 'mycluster1bda09' succeeded
NAME=bds_orcl_mycluster1
TYPE=generic_application
TARGET=ONLINE
STATE=ONLINE on mycluster1bda09

bds-exa-install: patching view LOADER_DIR_OBJS
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_catcon_30123.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon*.log files
for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon_*.lst
files for spool files, if any
```

```
catcon.pl: completed successfully
bds-exa-install: creating mta dblinks
bds-exa-install: cluster name           : mycluster1
bds-exa-install: extproc sid            : bds_orcl_mycluster1
bds-exa-install: cdb                   : true
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_dbcluster_dropdblink_catcon_30153.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_dbcluster_dropdblink*.log files for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_dbcluster_dropdblink_*.lst files for spool files, if any
catcon.pl: completed successfully
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_default_dropdblink_catcon_30179.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_default_dropdblink*.log files for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_default_dropdblink_*.lst files for spool files, if any
catcon.pl: completed successfully
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_dbcluster_createdblink_catcon_30205.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_dbcluster_createdblink*.log files for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_dbcluster_createdblink_*.lst files for spool files, if any
catcon.pl: completed successfully
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_default_createdblink_catcon_30231.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_default_createdblink*.log files for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/
bdscatcon_default_createdblink_*.lst files for spool files, if any
catcon.pl: completed successfully
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_catcon_30257.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon*.log files
for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon_*.lst
files for spool files, if any
catcon.pl: completed successfully
catcon: ALL catcon-related output will be written to /u03/app/oracle/product/
12.1.0/dbhome_1/install/bdscatcon_catcon_30283.lst
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon*.log files
for output generated by scripts
catcon: See /u03/app/oracle/product/12.1.0/dbhome_1/install/bdscatcon_*.lst
files for spool files, if any
catcon.pl: completed successfully
bds-exa-install: setup script completed all steps
```

For additional details see [“Running the bds-exa-install Script”](#).

7. Repeat step 6 for each database instance, if you have a multi instance database.

When the script completes, the following items including Oracle Big Data SQL is available and running on the database instance. However, if events cause the Oracle Big Data SQL agent to stop, then you must restart it. See [“Starting and Stopping the Big Data SQL Agent”](#).

- Oracle Big Data SQL directory and configuration with jar, and environment and properties files.

- Database dba_directories.
- Database dblinks.
- Database big data spfile parameter.

For example, you can verify the dba_directories from the SQL prompt as follows:

```
SQL> select * from dba_directories where directory_name like '%BIGDATA%';
```

2.2.2.1 Running the bds-exa-install Script

The bds-exa-install script generates a custom installation script that is run by the owner of the Oracle home directory. That secondary script installs all the files need by Oracle Big Data SQL into the \$ORACLE_HOME/bigdatasql directory. For Oracle NoSQL Database support, it installs the client library (kvclient.jar). It also creates the database directory objects, and the database links for the multithreaded Oracle Big Data SQL agent.

2.2.2.2 bds-exa-install Syntax

The following is the bds-exa-install syntax:

```
Usage: bds-exa-install oracle-sid=<orcl>
(
    --version
    --info
    --root-script-only
    --uninstall-as-primary
    --uninstall-as-secondary
    --install-as-secondary
    --jdk-home=<dir>
    --grid-home=<dir>
)*
```

Options

```
--version
    Prints script version.
--info
    Print information such as cluster name, CM host, Oracle Big Data Appliance
    HTTP server.
--root-script-only.
    Only generate the root script.
--uninstall-as-primary
    Uninstall scaj3lcdh, including hadoop client jars.
    Note: after this any secondary clusters needs to be reinstalled.
--uninstall-as-secondary
    Attempt to uninstall scaj3lcdh as a secondary cluster.
--install-as-secondary
    Default = false.
    Do not install client libraries, etc. The primary cluster will not be affected.
--jdk-home
    For example: /opt/oracle/bd_cell12.1.2.0.100_LINUX.X64_150912.1/jdk
--grid-home
    Oracle Grid Infrastructure home.
    For example: "/opt/oracle/bd_cell12.1.2.0.100_LINUX.X64_150912.1/./grid"
```

2.2.2.3 Troubleshooting Running bds-exa-install Script

In case of problems running the install script on Exadata, perform the following steps and open an SR with Oracle Support with the details:

1. Collect the debug output by running the script in a debug mode as follows:

```
$ ./bds-exa-install.sh --db-name=<dbresource> --grid-home=<grid home> --root-script=false --debug
OR
$ ./bds-exa-install.sh --root-script=false --debug
```

2. Collect the Oracle Database version as follows:

- a. Collect the result of `opatch lsinventory` from RDBMS-RAC Home.
- b. Collect the result of `opatch lsinventory` from Grid Home

3. Result of the following SQL statement to confirm that the Datapatch is set up.

```
SQL> select patch_id, patch_uid, version, bundle_series, bundle_id, action,
status from dba_registry_sqlpatch;
```

4. Collect the information from the following environment variables:

- \$ORACLE_HOME
- \$ORACLE_SID
- \$GI_HOME
- \$TNS_ADMIN

5. Result of running `lsnrctl status` command.

2.2.3 About Data Security with Oracle Big Data SQL

Oracle Big Data Appliance already provides numerous security features to protect data stored in a CDH cluster on Oracle Big Data Appliance:

- **Kerberos authentication:** Requires users and client software to provide credentials before accessing the cluster.
- **Apache Sentry authorization:** Provides fine-grained, role-based authorization to data and metadata.
- **HDFS Transparent Encryption:** Protects the data on disk and at rest. Data encryption and decryption is transparent to applications using the data.
- **Oracle Audit Vault and Database Firewall monitoring:** The Audit Vault plug-in on Oracle Big Data Appliance collects audit and logging data from MapReduce, HDFS, and Oozie services. You can then use Audit Vault Server to monitor these services on Oracle Big Data Appliance

Oracle Big Data SQL adds the full range of Oracle Database security features to this list. You can apply the same security policies and rules to your Hadoop data that you apply to your relational data.

2.2.4 Enabling Oracle Big Data SQL Access to a Kerberized Cluster

In order to give Oracle Big Data SQL access to HDFS data on a Kerberos-enabled cluster, make each Oracle Exadata Database Machine that needs access a Kerberos client. Also run `kinit` on the `oracle` account on each cluster node and Exadata Database Machine to ensure that the account is authenticated by Kerberos. There are two situations where this procedure is required:

- When enabling Oracle Big Data SQL on a Kerberos-enabled cluster.
- When enabling Kerberos on a cluster where Oracle Big Data SQL is already installed.

Note: Oracle Big Data SQL queries will run on the Hadoop cluster as the owner of the Oracle Database process (i.e. the `oracle` user). Therefore, the `oracle` user needs a valid Kerberos ticket in order to access data. This ticket is required for every Oracle Database instance that is accessing the cluster. A valid ticket is also need for each Big Data SQL Server process running on the Oracle Big Data Appliance. Run `kinit oracle` to obtain the ticket.

These steps enable the operating system user to authenticate with the **kinit** utility before submitting Oracle SQL Connector for HDFS jobs. The **kinit** utility typically uses a Kerberos keytab file for authentication without an interactive prompt for a password.

1. On each node of the cluster:
 - a. Log in as the `oracle` user.
 - b. Run `kinit` on the `oracle` account.

```
$ kinit oracle
```
 - c. Enter the Kerberos password.
2. Log on to the primary node and then stop and restart Oracle Big Data SQL.

```
$ bdaccli stop big_data_sql_cluster
$ bdaccli start big_data_sql_cluster
```
3. On all Oracle Exadata Database Machines that need access to the cluster:
 - a. Copy the Kerberos configuration file `/etc/krb5.conf` from the node where Mammoth is installed to the same path on each Oracle Exadata Machine.
 - b. Run `kinit` on the `oracle` account and enter the Kerberos password.
 - c. Re-run the Oracle Big Data SQL post-installation script

```
$ ./bds-exa-install.sh
```

Avoiding Kerberos Ticket Expiration

The system should run **kinit** on a regular basis, before letting the Kerberos ticket expire, to enable Oracle SQL Connector for HDFS to authenticate transparently. Use **cron** or a similar utility to run **kinit**. For example, if Kerberos tickets expire every two weeks, then set up a **cron** job to renew the ticket weekly.

2.2.5 Starting and Stopping the Big Data SQL Agent

The Big Data SQL agent on the database is managed by Oracle Clusterware. The agent is registered with Oracle Clusterware during Big Data SQL installation to automatically start and stop with the database. To check the status, you can run `mtactl check` from the Oracle Grid Infrastructure home or Oracle Clusterware home:

```
# mtactl check bds_databasename_clustername
```

2.3 Installing Oracle Big Data SQL on Other Hadoop Systems

Oracle Big Data SQL is deployed using the services provided by the cluster management server. The installation process uses the management server API to register the service and start the deployment task. From there, the management server controls the process.

After installing Big Data SQL on the cluster management server, use the tools provided in the bundle to generate an installation package for the database server side.

2.3.1 Downloading Oracle Big Data SQL

You can download Oracle Big Data SQL from the [Oracle Software Delivery Cloud](#)

1. On the cluster management server, create a new directory or choose an existing one to be the installation source directory.
2. Log in to the [Oracle Software Delivery Cloud](#).
3. Search for Oracle Big Data SQL.
4. Select Oracle Big Data SQL 3.0.0.0.0 for Linux x86-64.
5. Read and agree to the Oracle Standard Terms and Restrictions.
6. From the list of three files, select only one – the installer that is appropriate for your Hadoop system. You do not need V137415-01.zip.

```
Big Data SQL (3.0)
  V137415-01.zip Oracle Big Data SQL 3.0.0 cell software
only                                     440.6 MB
  V137419-01.zip Oracle Big Data SQL 3.0.0 installer for Cloudera
Enterprise                             625.4 MB
  V137420-01.zip Oracle Big Data SQL 3.0.0 installer for Hortonworks Data
Platform                               625.4 MB
```

7. Download the file and extract the contents.

Your product bundle should include the content listed in the table below.

Table 2-1 Oracle Big Data SQL Product Bundle Inventory

File	Description
setup-bds	Cluster-side installation script
bds-config.json	Configuration file.
api_env.sh	Setup REST API environment script
platform_env.sh	BDS service configuration script
BIGDATASQL-1.0.jar	CSD file (in the Cloudera product bundle only)
bin/json-select	JSON-select utility
db/bds-database-create-bundle.sh	Database bundle creation script

Table 2-1 (Cont.) Oracle Big Data SQL Product Bundle Inventory

db/database- install.zip	Database side installation files
repo/BIGDATASQL-1.0.0- el6.parcel	Parcel file (in the CDH product bundle only)
repo/manifest.json	Hash key for the parcel file (in the CDH product bundle only)
BIGDATASQL-1.0.0- el6.stack	Stack file (in the HDP product bundle only)
setup-db.sh	Script to acquire cluster information (Currently used in the manual portion of the HDP cluster-side installation.)

2.3.2 Prerequisites for Installing on an HDP Cluster

The following are required in order to install Oracle Big Data SQL on the Hortonworks Hadoop Data Platform (HDP).

Services Running

The following services must be running at the time of the Big Data SQL installation

- HDP 2.3
- Ambari 2.1.0
- HDFS 2.7.1
- YARN 2.7.1
- Zookeeper 3.4.6
- Hive 1.2.1
- Tez 0.7.0

Packages

The following packages must be pre-installed before installing Big Data SQL.

- JDK version 1.7 or later
- Python version 2.6.
- OpenSSL version 1.01 build 16 or later

System Tools

- curl
- rpm
- scp
- tar
- unzip

- `wget`
- `yum`

Environment Settings

The following environment settings are required prior to the installation.

- `ntp` enabled
- `iptables` disabled
- Ensure that `/usr/java/default` exists and is linked to the appropriate Java version. To link it to the latest Java version, perform the following as `root` :

```
$ ln -s /usr/java/latest /usr/java/default
```

Access Control Settings

The following user and group must exist.

- `oracle` user
- `oinstall` group

The `oracle` user must be a member of the `oinstall` group.

If Oracle Big Data SQL is Already Installed

If the Ambari Web GUI shows that Big Data SQL service is already installed, make sure that all Big Data SQL Cell components are stopped before reinstalling. (Use the **actions** button, as with any other service.)

2.3.3 Prerequisites for Installing on a CDH Cluster

The following conditions must be met when installing Oracle Big Data SQL on a CDH cluster that is not part of an Oracle Big Data Appliance.

Note: The installation prerequisites as well as the procedure for installing Oracle Big Data SQL on the Oracle Big Data Appliance are different from process used for installations on other CDH systems. See [Installing On Oracle Big Data Appliance and the Oracle Exadata Database Machine](#) if you are installing on Oracle Big Data Appliance.

Services Running

The following services must be running at the time of the Oracle Big Data SQL installation

- Cloudera's Distribution including Apache Hadoop (CDH) 5.5 and higher
- HDFS 2.6.0
- YARN 2.6.0
- Zookeeper 3.4.5
- Hive 1.1.0

Packages

The following packages must be pre-installed before installing Oracle Big Data SQL. The Oracle clients are available for download on the [Oracle Technology Network](#).

- JDK version 1.7 or later
- Oracle Instant Client – 12.1.0.2 or higher, e.g. oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64.rpm
- Oracle Instant JDBC Client – 12.1.0.2 or higher, e.g. oracle-instantclient12.1-jdbc-12.1.0.2.0-
- PERL LibXML – 1.7.0 or higher, e.g. perl-XML-LibXML-1.70-5.el6.x86_64.rpm
- Apache log4j

System Tools

- unzip
- finger
- wget

Environment Settings

The following environment settings are required prior to the installation.

- Ensure that `/usr/java/default` exists and is linked to the appropriate Java version. To link it to the latest Java version, perform the following as root :

```
$ ln -s /usr/java/latest /usr/java/default
```
- The path to the Java binaries must exist in `/usr/java/latest`.
- The default path to Hadoop libraries must be in `/opt/cloudera/parcels/CDH/lib/`.

Access Control Settings

The following user and group must exist.

- oracle user
- oinstall group

The oracle user must be a member of the oinstall group.

Settings to Save Before the Installation

If resource management is enabled on Cloudera Manager, then before installing Big Data SQL, save YARN's resource management configuration so that it can be restored to the original state if Big Data SQL is uninstalled later.

2.3.4 Installation Overview

The Oracle Big Data SQL installation consists of two stages.

- Cluster-side installation:
 - Deploys binaries along the cluster.

- Configures Linux and network settings for the service on each cluster node.
- Configures the service on the management server.
- Acquires cluster information for configure database connection.
- Creates database bundle for the database side installation.
- Oracle Database server-side installation:
 - Copies binaries into database node.
 - Configures network settings for the service.
 - Inserts cluster metadata into database.

2.3.5 Installing on the Hadoop Cluster Management Server

The first step of the Oracle Big Data SQL installation is to run the installer on the Hadoop cluster management server (where Cloudera Manager runs on a CDH system or Ambari runs on an HDP system). As post-installation task on the management server, you then run the script that prepares the installation bundle for the database server.

There are three tasks to perform on the cluster manager server:

- Extract the files from BIGDATASQL product bundle saved from the download (either `BigDataSQL-CDH-<version>.zip` or `BigDataSQL-HDP-<version>.zip`) then configure and run the Oracle Big Data SQL installer found within the bundle. This installs Oracle Big Data SQL on the local server.
- Run the database bundle creation script. This script generates the database bundle file that you will run on the Oracle Database server in order to install Oracle Big Data SQL there.
- Check the parameters in the database bundle file and adjust as needed.

After you have checked and (if necessary) edited the database bundle file, copy it over to the Oracle Database server and run it as described in [Installing on the Oracle Database Server](#)

Install Big Data SQL on the Cluster Management Server

To install Big Data SQL on the cluster management server:

1. Copy the appropriate zip file (`BigDataSQL-CDH-<version>.zip` or `BigDataSQL-HDP-<version>.zip`) to a temporary location on the cluster management server.
2. Unzip file.
3. Change directories to either `BigDataSQL-HDP-<version>` or `BigDataSQL-CDH-<version>`, depending up on which platform you are working with.
4. Edit the configuration file.

Table 2–4 below describes the use of each configuration parameter.

- For CDH, edit `bds-config.json`, as in this example. Any unused port will work as the web server port.

```
{
  "CLUSTER_NAME" : "cluster",
  "CSD_PATH" : "/opt/cloudera/csd",
  "DATABASE_IP" : "10.12.13.14/24",
  "REST_API_PORT" : "7180",
  "WEB_SERVER_PORT" : "81",
}
```

- For HDP, edit `bds-config.json` as in this example:

```
{
  "CLUSTER_NAME" : "clustername",
  "DATABASE_IP" : "10.10.10.10/24",
  "REST_API_PORT" : "8080",
}
```

`DATABASE_IP` must be the correct network interface address for the database node where you will perform the installation. You can confirm this by running `/sbin/ip -o -f inet addr show` on the database node.

5. Obtain the cluster administrator user ID and password and then as `root` run `setup-bds`. Pass it the configuration file name as an argument (`bds-config.json`). The script will prompt for the administrator credentials and then install BDS on the management server.

```
$ ./setup-bds bds-config.json
```

Table 2-2 Configuration Parameters for `setup-bds`

Configuration Parameter	Use	Applies To
<code>CLUSTER_NAME</code>	The name of the cluster on the Hadoop server.	CDH, HDP
<code>CSD_PATH</code>	Location of Custom Service Descriptor files.	CDH only
<code>DATABASE_IP</code>	The IP address of the Oracle Database server that will make connection requests. The address must include the prefix length (as in 100.112.10.36/24). Although only one IP address is specified in the configuration file, it is possible to install the database-side software on multiple database servers by using a command line parameter to override <code>DATABASE_IP</code> at installation time. (See the description of <code>--ip-cell</code> in Table 2-6 .)	CDH, HDP
<code>REST_API_PORT</code>	The port where the cluster management server listens for requests.	CDH, HDP
<code>WEB_SERVER_PORT</code>	A port assigned temporarily to a repository for deployment tasks during installation. This can be any port where the assignment does not conflict with cluster operations.	CDH only.

Important: Be sure that the address provided for `DATABASE_IP` is the correct address of a network interface on the database server and is accessible from each DataNode of the Hadoop system, otherwise the installation will fail. You can test that the database IP replies to a ping from each DataNode. Also, currently the address string (including the prefix length) must be at least nine characters long.

If the Oracle Big Data SQL Service Immediately Fails

If Ambari or Configuration Manager reports an Oracle Big Data SQL service failure immediately after service startup, do the following.

1. Check the cell server (CELLSRV) log on the cluster management server for the following error at the time of failure:

```
ossnet_create_box_handle: failed to parse ip : <IP Address>
```

2. If the IP address in the error message is less than nine characters in length, for example, `10.0.1.4/24`, then on the cluster management server, find this address in `/opt/oracle/bd_cell/cellsrv/deploy/config/cellinit.ora`. Edit the string by padding one or more of the octets with leading zeros to make the total at least nine characters in length, as in:

```
ipaddress1=10.0.1.004/24
```

3. Restart the Oracle Big Data SQL service.

The need for this workaround will be eliminated in a subsequent Oracle Big Data SQL release.

2.3.6 Creating the Database-Side Installation Bundle

On the cluster management server, run the database bundle creation script from the Oracle Big Data SQL download to create an installation bundle to install the product on the Oracle Database server. If some of the external resources that the script requires are not accessible from the management server, you can add them manually.

The database bundle creation script attempts to download the following:

- Hadoop and Hive client tarballs from Cloudera or Hortonworks repository web site.
- Configuration files for Yarn and Hive from the cluster management server, via Cloudera Manager (for the CDH versions) or Ambari (for the HDP versions).
- For HDP only, HDFS and MapReduce configuration files from Ambari.

1. Change directories to `BigDataSQL-CDH-<version>/db` or `(BigDataSQL-HDP-<version>/db)`.
2. Run the BDS database bundle creation script. See the table below for optional parameters that you can pass to the script in order to override any of the default settings.

```
$ bds-database-create-bundle.sh <optional parameters>
```

The message below is returned if the operation is successful.

bds-database-create-bundle: database bundle creation script completed all steps

The database bundle file includes a number of parameters. You can change any of these parameters as necessary. Any URLs specified must be accessible from the cluster management server at the time you run `bds-database-create-bundle.sh`.

Table 2-3 Command Line Parameters for `bds-database-create-bundle.sh`

Parameter	Value
<code>--hadoop-client-ws</code>	Specifies an URL for the Hadoop client tarball download or bypass download of this client.
<code>--no-hadoop-client-ws</code>	
<code>--hive-client-ws</code>	Specifies an URL for the Hive client tarball download or bypass download of this client.
<code>--no-hive-client-ws</code>	
<code>--yarn-conf-ws</code>	Specifies an URL for the YARN configuration zip file download or bypass this download.
<code>--no-yarn-conf-ws</code>	
<code>--hive-conf-ws</code>	Specifies an URL for the Hive configuration zip file download or bypass this download.
<code>--no-hive-conf-ws</code>	
<code>--ignore-missing-files</code>	Create the bundle file even if some files are missing.
<code>--clean-previous</code>	Deletes previous bundle files and directories from <code>bds-database-install/</code>
<code>--script-only</code>	Only creates the script database installation file.
<code>--hdfs-conf-ws</code>	Specify an URL for the HDFS configuration zip file download or bypass this download (HDP only).
<code>--no-hdfs-conf-ws</code>	
<code>--mapreduce-conf-ws</code>	Specify an URL for the MapReduce configuration zip file download or bypass this download (HDP only).
<code>--no-mapreduce-conf-ws</code>	

Note: In Big Data SQL 3.0, `bds-database-create-bundle.sh` does not include a command line parameter to override the default JDK (`jdk-8u66-linux-x64`). To include a different version of the JDK, run `bds-database-create-bundle.sh` twice, as follows.

1. Run `bds-database-create-bundle.sh` to generate some files that you will edit or replace.
 2. Remove the existing JDK file, `/bds_database_install/jdk-8u66-linux-x64.tar.gz`.
 3. Remove the `bds-database-install.zip` bundle file generated by this first run of the script.
 4. Manually download the JDK tarball from [Oracle Technology Network](#). Copy it into `/bds_database_install`.
 5. Edit `bds-database-install/db/create-bundle.env`. Update the `$jdktar` environment variable to match the JDK you downloaded. For example: `jdktar=jdk-8u77-linux-x64.tar.gz`
 6. Run `bds-database-create-bundle.sh` again to generate a new database bundle.
-

Manually Adding Resources if Download Sites are not Accessible to the BDS Database Bundle Creation Script

If one or more of the default download sites is inaccessible from the cluster management server, there are two ways around this problem:

- Download the files from another server first and then provide `bds-database-create-bundle.sh` with the alternate path as an argument. For example:

```
$ ./bds-database-create-bundle.sh --yarn-conf-ws='http://nodexample:1234/config/yarn'
```

- Because the script will first search locally in `/bds-database-install` for resources, you can download the files to another server, move the files into `/bds-database-install` on the cluster management server and then run the bundle creation script with no additional argument. For example:

```
$ cp hadoop-xxxx.tar.gz bds-database-install/
$ cp hive-xxxx.tar.gz bds-database-install/
$ cp yarn-conf.zip bds-database-install/
$ cp hive-conf.zip bds-database-install/
$ cd db
$ ./bds-database-create-bundle.sh
```

Copying the Database Bundle to the Oracle Database Server

Use `scp` to copy the database bundle you created to the Oracle Database server. In the example below, `dbnode` is the database server. The Linux account and target directory here are arbitrary. Use any account authorized to `scp` to the specified path.

```
$ scp bds-database-install.zip oracle@dbnode:/home/oracle
```

The next step is to log on to the Oracle Database server and install the bundle.

2.3.7 Installing on the Oracle Database Server

Oracle Big Data SQL must be installed on both the Hadoop cluster management server and the Oracle Database server. This section describes the database server installation.

The information in this section does not apply to the installation of Oracle Big Data SQL on an Oracle Exadata Database Machine connected Oracle Big Data Appliance.

Important:

For multi-node databases, you must repeat this installation on every node of the database. For each node, you may need to modify the `DATABASE_IP` parameter of the installation bundle in order to identify the correct network interface. This is described in the section, *If You Need to Change the Configured Database_IP Address*

Prerequisites for Installing on an Oracle Database Server

Required Software

See the *Oracle Big Data SQL Master Compatibility Matrix* (Doc ID 2119369.1) in [My Oracle Support](#) for supported Linux distributions, Oracle Database release levels, and required patches.

Note: Be sure that the correct Bundle Patch and one-off patch have been pre-applied before starting this installation. Earlier Bundle patches are not supported for use Big Data SQL 3.0 at this time.

Recommended Network Connections to the Hadoop Cluster

Oracle recommends Ethernet connections between Oracle Database and the Hadoop cluster of 10Gb/s Ethernet.

Extract and Run the Big Data SQL Installation Script

Perform the procedure in this section as the `oracle` user, except where `sudo` is indicated.

1. Check that `/etc/oracle/cell/network-config/cellinit.ora` exists. If not, do the following to create it and add the database server IP address:

- a. Create the network-config directory and set permissions:

```
$ sudo mkdir -p /etc/oracle/cell/network-config/
$ sudo chown oracle:dba /etc/oracle/cell/network-config
$ sudo chmod ug+wx /etc/oracle/cell/network-config
```

- b. Find the private IP address of the database server (through `inet addr` or other means)

The address must include the prefix length, as in `100.112.10.36/24`.

- c. Create `cellinit.ora` under `network-config` and add the IP address as the value of `ipaddress1` as shown below. Also be sure to include the two lines that follow.

```
ipaddress1=100.112.10.36/24
_skgxp_ant_options=1
_skgxp_dynamic_protocol=2
```

2. Locate the database bundle zip file that you copied over from the cluster management server.
3. Unzip the bundle into a temporary directory.
4. Change directories to `bds-database-install`, which was extracted from the zip file.
5. Run `bds-database-install.sh`. Note the optional parameters listed in [Table 2-4](#)

Table 2-4 *Optional Parameters for `bds-database-install.sh`*

Parameter	Function
<code>--version</code>	Show the <code>bds-database-install.sh</code> script version.
<code>--info</code>	Show information about the cluster.
<code>--ip-cell</code>	Set a particular IP address for <code>db_cell</code> process. See <i>If You Need to Change the Configured Database_IP Address</i> below
<code>--install-as-secondary</code>	Specify secondary cluster installation.
<code>--uninstall-as-primary</code>	Uninstall Oracle Big Data SQL from the primary cluster.
<code>--uninstall-as-secondary</code>	Uninstall Oracle Big Data SQL from a secondary cluster.
<code>--jdk-home</code>	Specify the JDK home directory.
<code>--grid-home</code>	Specify the Grid home directory.
<code>--db-name</code>	Specify the Oracle Database SID.
<code>--debug</code>	Activate shell trace mode. If you report a problem, Oracle Support may want to see this output.

If You Need to Change the Configured Database_IP Address

The `DATABASE_IP` parameter in the `bds-config.json` file identifies the network interface of the database node. If you run `bds-database-install.sh` with no parameter passed in, it will search for that IP address (with that length, specifically) among the available network interfaces. You can pass the `--ip-cell` parameter to `bds-database-install.sh` in order to override the configured `DATABASE_IP` setting:

```
$ ./bds-database-install.sh --ip-cell=10.20.30.40/24
```

Possible reasons for doing this are:

- `bds-database-install.sh` terminates with an error. The configured IP address (or length) may be wrong.

- There is an additional database node in the cluster and the defined DATABASE_IP address is not a network interface of the current node.
- The connection is to a multi-node database. In this case, perform the installation on each database node. On each node, use the `--ip-cell` parameter to set the correct DATABASE_IP value.

To determine the correct value for `ip-cell`, you can use list all network interfaces on a node as follows:

```
/sbin/ip -o -f inet addr show
```

2.3.8 Uninstalling Oracle Big Data SQL

The steps for uninstalling Oracle Big Data SQL from HDP and from CDH systems are different.

Uninstalling the Software from an HDP Hadoop Cluster

1. In the Ambari web interface, stop the Big Data SQL service. All components on all DataNodes must be stopped.
2. On the Ambari command line, delete the Big Data SQL service using a REST API call.

```
curl --user admin:admin -H 'X-Requested-By:<user>' -X DELETE http://
<ambari_server_fqdn>:<rest_api_port>/api/v1/clusters/<cluster_name>/services/
BIGDATASQL
```

3. On each DataNode, find and kill any Oracle Big Data SQL processes that are running.

```
# ps -fea | grep bds
# kill -9 <pid>
```

4. On the Ambari command line, remove the BIGDATASQL stack from the services.

```
# rm -rf /var/lib/ambari-server/resources/stacks/HDP/<version>/services/
BIGDATASQL
```

5. On each DataNode, remove the `bd_cell` RPM.

```
# yum remove -y bd_cell
```

6. On all DataNodes, remove the following directories.

```
# rm -rf /opt/oracle/bd_cell
# rm -rf /opt/oracle/bigdatasql
# rm -rf /tmp/bigdatasql
# rm -rf /var/log/oracle
```

7. On the Ambari command line, restart Ambari.

```
# ambari-server restart
```

Uninstalling Oracle Big Data SQL From a CDH Hadoop Cluster

To uninstall Big Data SQL from a CDH cluster (that is not hosted on Oracle Big Data Appliance), follow these steps:

1. In the Cloudera Manager GUI, do the following:

- a. Stop the Big Data SQL service. All instances on all DataNodes must be stopped.
 - b. Delete the service from the cluster.
 - c. Deactivate and remove the parcel from all hosts.
 - d. Delete the parcel.
 - e. If resource management was not enabled on Cloudera Manager before Big Data SQL installation, disable the option **Cgroup-based Resource Management** and restart the YARN service.
2. On each DataNode, kill any bds processes.

```
# ps -fea | grep bds  
# kill -9 <pid>
```
3. On the Cloudera Manager command line, remove the Big Data SQL JAR from the `csd` directory. (The default location is `/opt/cloudera/csd`, but this may differ.)

```
# rm -f /opt/cloudera/csd/BIGDATASQL-1.0.jar
```
4. On each DataNode, do the following:
 - a. Remove the `bd_cell` RPM.

```
# yum remove -y bd_cell
```
 - b. Remove all Oracle Big Data SQL directories.

```
# rm -rf /opt/oracle/bd_cell  
# rm -rf /opt/oracle/bigdatasql  
# rm -rf /tmp/bigdatasql  
# rm -rf /var/log/oracle
```

2.3.9 Securing Big Data SQL

Procedures for securing Oracle Big Data SQL on Hortonworks HDP and on CDH-based systems other than Oracle Big Data Appliance are not covered in this version of the guide. Please review the MOS documents referenced in this section for more information.

2.3.9.1 Big Data SQL Communications and Secure Hadoop Clusters

Please refer to MOS Document 2123125.1 at [My Oracle Support](#) for guidelines on securing Hadoop clusters for use with Big Data SQL.

2.3.9.2 Setting up Oracle Big Data SQL and Oracle Secure External Password Store

See MOS Document 2126903.1 for changes required in order to use Oracle Secure External Password Store with Oracle Big Data SQL

Using Oracle Big Data SQL for Data Access

This chapter describes how to use Oracle Big Data SQL to create external tables and access data from Hadoop data sources as well as Oracle NoSQL Database.

It also describes some of the changes that Oracle Big Data SQL makes on the Oracle Database server.

- [Creating an Oracle External Table for Hive Data](#)
- [Creating an Oracle External Table for Oracle NoSQL Database](#)
- [Creating an Oracle External Table for Apache HBase](#)
- [Creating an Oracle External Table for HDFS Files](#)
- [About the SQL CREATE TABLE Statement](#)
- [About Data Type Conversions](#)
- [Querying External Tables](#)
- [About Oracle Big Data SQL on the Database Server \(Oracle Exadata Machine or Other\)](#)

3.1 Creating an Oracle External Table for Hive Data

You can easily create an Oracle external table for data in Apache Hive. Because the metadata is available to Oracle Database, you can query the data dictionary for information about Hive tables. Then you can use a PL/SQL function to generate a basic SQL `CREATE TABLE EXTERNAL ORGANIZATION` statement. You can modify the statement before execution to customize the external table.

3.1.1 Obtaining Information About a Hive Table

The `DBMS_HADOOP` PL/SQL package contains a function named `CREATE_EXTDDL_FOR_HIVE`. It returns the data dictionary language (DDL) to create an external table for accessing a Hive table. This function requires you to provide basic information about the Hive table:

- Name of the Hadoop cluster
- Name of the Hive database
- Name of the Hive table
- Whether the Hive table is partitioned

You can obtain this information by querying the ALL_HIVE_TABLES data dictionary view. It displays information about all Hive tables that you can access from Oracle Database.

This example shows that the current user has access to an unpartitioned Hive table named RATINGS_HIVE_TABLE in the default database. A user named JDOE is the owner.

```
SQL> SELECT cluster_id, database_name, owner, table_name, partitioned FROM
all_hive_tables;
CLUSTER_ID    DATABASE_NAME  OWNER      TABLE_NAME      PARTITIONED
-----
hadoop1       default       jdoe       ratings_hive_table UN-PARTITIONED
```

See Also:

[“Static Data Dictionary Views for Hive”](#)

3.1.2 Using the CREATE_EXTDDL_FOR_HIVE Function

With the information from the data dictionary, you can use the CREATE_EXTDDL_FOR_HIVE function of DBMS_HADOOP. This example specifies a database table name of RATINGS_DB_TABLE in the current schema. The function returns the text of the CREATE TABLE command in a local variable named DDLout, but does not execute it.

```
DECLARE
    DDLout VARCHAR2(4000);
BEGIN
    dbms_hadoop.create_extddl_for_hive(
        CLUSTER_ID=>'hadoop1',
        DB_NAME=>'default',
        HIVE_TABLE_NAME=>'ratings_hive_table',
        HIVE_PARTITION=>FALSE,
        TABLE_NAME=>'ratings_db_table',
        PERFORM_DDL=>FALSE,
        TEXT_OF_DDL=>DDLout
    );
    dbms_output.put_line(DDLout);
END;
/
```

When this procedure runs, the PUT_LINE function displays the CREATE TABLE command:

```
CREATE TABLE ratings_db_table (
    c0 VARCHAR2(4000),
    c1 VARCHAR2(4000),
    c2 VARCHAR2(4000),
    c3 VARCHAR2(4000),
    c4 VARCHAR2(4000),
    c5 VARCHAR2(4000),
    c6 VARCHAR2(4000),
    c7 VARCHAR2(4000))
ORGANIZATION EXTERNAL
    (TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
    ACCESS PARAMETERS
        (
            com.oracle.bigdata.cluster=hadoop1
```



```

        com.oracle.bigdata.tablename=default.ratings_hive_table
    )
) PARALLEL 2 REJECT LIMIT UNLIMITED

```

You can capture this information in a SQL script, and use the access parameters to change the Oracle table name, the column names, and the data types as desired before executing it. You might also use access parameters to specify a date format mask.

The ALL_HIVE_COLUMNS view shows how the default column names and data types are derived. This example shows that the Hive column names are C0 to C7, and that the Hive STRING data type maps to VARCHAR2(4000):

```
SQL> SELECT table_name, column_name, hive_column_type, oracle_column_type FROM
all_hive_columns;
```

TABLE_NAME	COLUMN_NAME	HIVE_COLUMN_TYPE	ORACLE_COLUMN_TYPE
ratings_hive_table	c0	string	VARCHAR2(4000)
ratings_hive_table	c1	string	VARCHAR2(4000)
ratings_hive_table	c2	string	VARCHAR2(4000)
ratings_hive_table	c3	string	VARCHAR2(4000)
ratings_hive_table	c4	string	VARCHAR2(4000)
ratings_hive_table	c5	string	VARCHAR2(4000)
ratings_hive_table	c6	string	VARCHAR2(4000)
ratings_hive_table	c7	string	VARCHAR2(4000)

8 rows selected.

See Also:

[“DBMS_HADOOP PL/SQL Package”](#)

3.1.3 Developing a CREATE TABLE Statement for ORACLE_HIVE

You can choose between using DBMS_HADOOP and developing a CREATE TABLE statement from scratch. In either case, you may need to set some access parameters to modify the default behavior of ORACLE_HIVE.

3.1.3.1 Using the Default ORACLE_HIVE Settings

The following statement creates an external table named ORDER to access Hive data:

```

CREATE TABLE order (cust_num    VARCHAR2(10),
                    order_num    VARCHAR2(20),
                    description  VARCHAR2(100),
                    order_total  NUMBER (8,2))
    ORGANIZATION EXTERNAL (TYPE oracle_hive);

```

Because no access parameters are set in the statement, the ORACLE_HIVE access driver uses the default settings to do the following:

- Connects to the default Hadoop cluster.
- Uses a Hive table named order. An error results if the Hive table does not have fields named CUST_NUM, ORDER_NUM, DESCRIPTION, and ORDER_TOTAL.
- Sets the value of a field to NULL if there is a conversion error, such as a CUST_NUM value longer than 10 bytes.

3.1.3.2 Overriding the Default ORACLE_HIVE Settings

You can set properties in the `ACCESS PARAMETERS` clause of the external table clause, which override the default behavior of the access driver. The following clause includes the `com.oracle.bigdata.overflow` access parameter. When this clause is used in the previous example, it truncates the data for the `DESCRIPTION` column that is longer than 100 characters, instead of throwing an error:

```
(TYPE oracle_hive
 ACCESS PARAMETERS (
   com.oracle.bigdata.overflow={"action":"truncate", "col":"DESCRIPTION"} ))
```

The next example sets most of the available parameters for `ORACLE_HIVE`:

```
CREATE TABLE order (cust_num VARCHAR2(10),
                     order_num VARCHAR2(20),
                     order_date DATE,
                     item_cnt NUMBER,
                     description VARCHAR2(100),
                     order_total (NUMBER(8,2)) ORGANIZATION EXTERNAL
 (TYPE oracle_hive
  ACCESS PARAMETERS (
    com.oracle.bigdata.tablename: order_db.order_summary
    com.oracle.bigdata.colmap:    {"col":"ITEM_CNT", \
                                   "field":"order_line_item_count"}
    com.oracle.bigdata.overflow: {"action":"TRUNCATE", \
                                   "col":"DESCRIPTION"}
    com.oracle.bigdata.erroropt: [{"action":"replace", \
                                   "value":"INVALID_NUM" , \
                                   "col":["CUST_NUM","ORDER_NUM"]} ,\
                                   {"action":"reject", \
                                   "col":"ORDER_TOTAL"}
  ))
```

The parameters make the following changes in the way that the `ORACLE_HIVE` access driver locates the data and handles error conditions:

- `com.oracle.bigdata.tablename`: Handles differences in table names. `ORACLE_HIVE` looks for a Hive table named `ORDER_SUMMARY` in the `ORDER.DB` database.
- `com.oracle.bigdata.colmap`: Handles differences in column names. The Hive `ORDER_LINE_ITEM_COUNT` field maps to the Oracle `ITEM_CNT` column.
- `com.oracle.bigdata.overflow`: Truncates string data. Values longer than 100 characters for the `DESCRIPTION` column are truncated.
- `com.oracle.bigdata.erroropt`: Replaces bad data. Errors in the data for `CUST_NUM` or `ORDER_NUM` set the value to `INVALID_NUM`.

3.2 Creating an Oracle External Table for Oracle NoSQL Database

You can use the `ORACLE_HIVE` access driver to access data stored in Oracle NoSQL Database. However, you must first create a Hive external table that accesses the `KVStore`. Then you can create an external table in Oracle Database over it, similar to the process described in [“Creating an Oracle External Table for Hive Data”](#).

This section contains the following topics:

- [Creating a Hive External Table for Oracle NoSQL Database](#)
- [Creating the Oracle Database Table for Oracle NoSQL Data](#)
- [About Column Data Type Mappings](#)
- [Example of Accessing Data in Oracle NoSQL Database](#)

3.2.1 Creating a Hive External Table for Oracle NoSQL Database

To provide access to the data in Oracle NoSQL Database, you create a Hive external table over the Oracle NoSQL table. Oracle Big Data SQL provides a StorageHandler named `oracle.kv.hadoop.hive.table.TableStorageHandler` that enables Hive to read the Oracle NoSQL Database table format.

The following is the basic syntax of a Hive `CREATE TABLE` statement for a Hive external table over an Oracle NoSQL table:

```
CREATE EXTERNAL TABLE tablename colname coltype[, colname coltype,...]
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
    "oracle.kv.kvstore" = "database",
    "oracle.kv.hosts" = "nosql_node1:port[, nosql_node2:port...]",
    "oracle.kv.hadoop.hosts" = "hadoop_node1[,hadoop_node2...]",
    "oracle.kv.tableName" = "table_name");
```

Hive `CREATE TABLE` Parameters

tablename

The name of the Hive external table being created.

This table name will be used in SQL queries issued in Oracle Database, so choose a name that is appropriate for users. The name of the external table that you create in Oracle Database must be identical to the name of this Hive table.

Table, column, and field names are case insensitive in Oracle NoSQL Database, Apache Hive, and Oracle Database.

colname coltype

The names and data types of the columns in the Hive external table. See [Table 3-1](#) for the data type mappings between Oracle NoSQL Database and Hive.

Hive `CREATE TABLE TBLPROPERTIES` Clause

oracle.kv.kvstore

The name of the KVStore. Only upper- and lowercase letters and digits are valid in the name.

oracle.kv.hosts

A comma-delimited list of host names and port numbers in the Oracle NoSQL Database cluster. Each string has the format *hostname:port*. Enter multiple names to provide redundancy in the event that a host fails.

oracle.kv.hadoop.hosts

A comma-delimited list of all host names in the Hadoop cluster with Oracle Big Data SQL enabled.

oracle.kv.tableName

The name of the table in Oracle NoSQL Database that stores the data for this Hive external table.

See Also:

Apache Hive Language Manual DDL at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>

3.2.2 Creating the Oracle Database Table for Oracle NoSQL Data

Use the following syntax to create an external table in Oracle Database that can access the Oracle NoSQL data through a Hive external table:

```
CREATE TABLE tablename(colname colType[, colname colType...])
  ORGANIZATION EXTERNAL
  (TYPE ORACLE_HIVE DEFAULT DIRECTORY directory
  ACCESS PARAMETERS
    (access parameters)
  )
  REJECT LIMIT UNLIMITED;
```

In this syntax, you identify the column names and data types. For more about this syntax, see “[About the SQL CREATE TABLE Statement](#)”.

3.2.3 About Column Data Type Mappings

When Oracle Big Data SQL retrieves data from Oracle NoSQL Database, the data is converted twice to another data type:

- To a Hive data type when the data is read into the columns of the Hive external table.
- To an Oracle data type when the data is read into the columns of an Oracle Database external table.

[Table 3-1](#) identifies the supported Oracle NoSQL data types and their mappings to Hive and Oracle Database data types. Oracle Big Data SQL does not support the Oracle NoSQL complex data types Array, Map, and Record.

Table 3-1 Oracle NoSQL Database Data Type Mappings

Oracle NoSQL Database Data Type	Apache Hive Data Type	Oracle Database Data Type
String	STRING	VARCHAR2
Boolean	BOOLEAN	NUMBER ¹
Integer	INT	NUMBER
Long	INT	NUMBER
Double	DOUBLE	NUMBER(<i>p,s</i>)

Table 3-1 (Cont.) Oracle NoSQL Database Data Type Mappings

Oracle NoSQL Database Data Type	Apache Hive Data Type	Oracle Database Data Type
Float	FLOAT	NUMBER(<i>p,s</i>)

¹ 0 for false, and 1 for true

3.2.4 Example of Accessing Data in Oracle NoSQL Database

This example uses the sample data provided with the Oracle NoSQL Database software:

- [Creating the Oracle NoSQL Database Example Table](#)
- [Creating the Example Hive Table for vehicleTable](#)
- [Creating the Oracle Table for VEHICLES](#)

3.2.4.1 Creating the Oracle NoSQL Database Example Table

Verify that the following files reside in the `examples/hadoop/table` directory:

```
create_vehicle_table.kvs
CountTableRows.java
LoadVehicleTable.java
```

This example runs on a Hadoop cluster node named `some1node07` and uses a KVStore named `SOME1KV`.

To create and populate the sample table in Oracle NoSQL Database:

1. Open a connection to an Oracle NoSQL Database node on your Hadoop cluster.
2. Create a table named `vehicleTable`. The following example uses the `load` command to run the commands in `create_vehicle_table.kvs`:

```
$ cd NOSQL_HOME
$ java -jar lib/kvcli.jar -host some1node07 -port 5000 \
  load -file examples/hadoop/table/create_vehicle_table.kvs
```

3. Compile `LoadVehicleTable.java`:

```
$ javac -cp examples:lib/kvclient.jar examples/hadoop/table/LoadVehicleTable.java
```

4. Execute the `LoadVehicleTable` class to populate the table:

```
$ java -cp examples:lib/kvclient.jar hadoop.table.LoadVehicleTable -host
some1node07 -port 5000 -store SOME1KV
{"type":"auto","make":"Chrysler","model":"PTCruiser","class":"4WheelDrive","color":"white","price":20743.240234375,"count":30}
{"type":"suv","make":"Ford","model":"Escape","class":"FrontWheelDrive","color":"
.
.
.
10 new records added
```

The `vehicleTable` table contains the following fields:

Field Name	Data Type
type	STRING
make	STRING
model	STRING
class	STRING
color	STRING
price	DOUBLE
count	INTEGER

3.2.4.2 Creating the Example Hive Table for vehicleTable

The following example creates a Hive table named `VEHICLES` that accesses `vehicleTable` in the `SOME1KV` KVStore. In this example, the system is configured with a Hadoop cluster in the first six servers (`some1node01` to `some1node06`) and an Oracle NoSQL Database cluster in the next three servers (`some1node07` to `some1node09`).

```
CREATE EXTERNAL TABLE IF NOT EXISTS vehicles
(
  type STRING,
  make STRING,
  model STRING,
  class STRING,
  color STRING,
  price DOUBLE,
  count INT)
COMMENT 'Accesses data in vehicleTable in the SOME1KV KVStore'
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES
(
  "oracle.kv.kvstore" = "SOME1KV",
  "oracle.kv.hosts" = "some1node07.example.com:5000,some1node08.example.com:5000",
  "oracle.kv.hadoop.hosts" =
"some1node01.example.com,some1node02.example.com,some1node03.example.com,some1node04.
example.com,some1node05.example.com,some1node06.example.com",
  "oracle.kv.tableName" = "vehicleTable");
```

The `DESCRIBE` command lists the columns in the `VEHICLES` table:

```
hive> DESCRIBE vehicles;
OK
type                string                from deserializer
make                string                from deserializer
model               string                from deserializer
class               string                from deserializer
color               string                from deserializer
price               double               from deserializer
count               int                  from deserializer
```

A query against the Hive `VEHICLES` table returns data from the Oracle NoSQL `vehicleTable` table:

```
hive> SELECT make, model, class
FROM vehicletable
```

```

WHERE type='truck' AND color='red'
ORDER BY make, model;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
.
.
.
Chrysler      Ram1500      RearWheelDrive
Chrysler      Ram2500      FrontWheelDrive
Ford          F150        FrontWheelDrive
Ford          F250        RearWheelDrive
Ford          F250        AllWheelDrive
Ford          F350        RearWheelDrive
GM            Sierra       AllWheelDrive
GM            Silverado1500 RearWheelDrive
GM            Silverado1500 AllWheelDrive

```

3.2.4.3 Creating the Oracle Table for VEHICLES

After you create the Hive table, the metadata is available in the Oracle Database static data dictionary views. The following SQL `SELECT` statement returns information about the Hive table created in the previous topic:

```

SQL> SELECT table_name, column_name, hive_column_type
      FROM all_hive_columns
      WHERE table_name='vehicles';

```

TABLE_NAME	COLUMN_NAME	HIVE_COLUMN_TYPE
vehicles	type	string
vehicles	make	string
vehicles	model	string
vehicles	class	string
vehicles	color	string
vehicles	price	double
vehicles	count	int

The next SQL `CREATE TABLE` statement generates an external table named `VEHICLES` over the Hive `VEHICLES` table, using the `ORACLE_HIVE` access driver. The name of the table in Oracle Database must be identical to the name of the table in Hive. However, both Oracle NoSQL Database and Oracle Database are case insensitive.

```

CREATE TABLE vehicles
(type VARCHAR2(10), make VARCHAR2(12), model VARCHAR2(20),
 class VARCHAR2(40), color VARCHAR2(20), price NUMBER(8,2),
 count NUMBER)
ORGANIZATION EXTERNAL
(TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
 ACCESS PARAMETERS
 (com.oracle.bigdata.debug=true com.oracle.bigdata.log.opt=normal))
REJECT LIMIT UNLIMITED;

```

This SQL `SELECT` statement retrieves all rows for red trucks from `vehicleTable` in Oracle NoSQL Database:

```

SQL> SELECT make, model, class
      FROM vehicles
      WHERE type='truck' AND color='red'
      ORDER BY make, model;

```

MAKE	MODEL	CLASS
-----	-----	-----

Chrysler	Ram1500	RearWheelDrive
Chrysler	Ram2500	FrontWheelDrive
Ford	F150	FrontWheelDrive
Ford	F250	AllWheelDrive
Ford	F250	RearWheelDrive
Ford	F350	RearWheelDrive
GM	Sierra	AllWheelDrive
GM	Silverado1500	RearWheelDrive
GM	Silverado1500	4WheelDrive
GM	Silverado1500	AllWheelDrive

3.3 Creating an Oracle External Table for Apache HBase

You can also use the `ORACLE_HIVE` access driver to access data stored in Apache HBase. However, you must first create a Hive external table that accesses the HBase table. Then you can create an external table in Oracle Database over it. The basic steps are the same as those described in “[Creating an Oracle External Table for Oracle NoSQL Database](#)”.

3.3.1 Creating a Hive External Table for HBase

To provide access to the data in an HBase table, you create a Hive external table over it. Apache provides a storage handler and a SerDe that enable Hive to read the HBase table format.

The following is the basic syntax of a Hive `CREATE TABLE` statement for an external table over an HBase table:

```
CREATE EXTERNAL TABLE tablename colname coltype[, colname coltype,...]
ROW FORMAT
  SERDE 'org.apache.hadoop.hive.hbase.HBaseSerDe'
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
  'serialization.format'='1',
  'hbase.columns.mapping'=':key,value:key,value:
```

See Also:

- *Apache Hive Language Manual DDL* at <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>
 - *Hive HBase Integration* at <https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration#HBaseIntegration-StorageHandlers>
 - Class `HBaseSerDe` in the Hive API reference at <http://hive.apache.org/javadocs/r0.13.1/api/hbase-handler/index.html>
-
-

3.3.2 Creating the Oracle Database Table for HBase

Use the following syntax to create an external table in Oracle Database that can access the HBase data through a Hive external table:


```
CREATE TABLE tablename(colname colType[, colname colType...])
  ORGANIZATION EXTERNAL
    (TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
      ACCESS PARAMETERS
        (access parameters)
    )
  REJECT LIMIT UNLIMITED;
```

In this syntax, you identify the column names and data types. To specify the access parameters, see [“About the SQL CREATE TABLE Statement”](#).

3.4 Creating an Oracle External Table for HDFS Files

The ORACLE_HDFS access driver enables you to access many types of data that are stored in HDFS, but which do not have Hive metadata. You can define the record format of text data, or you can specify a SerDe for a particular data format.

You must create the external table for HDFS files manually, and provide all the information the access driver needs to locate the data, and parse the records and fields. The following are some examples of CREATE TABLE ORGANIZATION EXTERNAL statements.

3.4.1 Using the Default Access Parameters with ORACLE_HDFS

The following statement creates a table named ORDER to access the data in all files stored in the /usr/cust/summary directory in HDFS:

```
CREATE TABLE ORDER (cust_num VARCHAR2(10),
                     order_num VARCHAR2(20),
                     order_total NUMBER (8,2))
  ORGANIZATION EXTERNAL
    ( TYPE oracle_hdfs
      DEFAULT DIRECTORY DEFAULT_DIR
    )
  LOCATION ('hdfs:/usr/cust/summary/*');
```

Because no access parameters are set in the statement, the ORACLE_HDFS access driver uses the default settings to do the following:

- Connects to the default Hadoop cluster.
- Reads the files as delimited text, and the fields as type STRING.
- Assumes that the number of fields in the HDFS files match the number of columns (three in this example).
- Assumes the fields are in the same order as the columns, so that CUST_NUM data is in the first field, ORDER_NUM data is in the second field, and ORDER_TOTAL data is in the third field.
- Rejects any records in which the value causes a data conversion error: If the value for CUST_NUM exceeds 10 characters, the value for ORDER_NUM exceeds 20 characters, or the value of ORDER_TOTAL cannot be converted to NUMBER.

3.4.2 Overriding the Default ORACLE_HDFS Settings

You can use many of the same access parameters with ORACLE_HDFS as ORACLE_HIVE.

3.4.2.1 Accessing a Delimited Text File

The following example is equivalent to the one shown in [“Overriding the Default ORACLE_HIVE Settings”](#). The external table access a delimited text file stored in HDFS.

```
CREATE TABLE order (cust_num VARCHAR2(10),
                    order_num VARCHAR2(20),
                    order_date DATE,
                    item_cnt NUMBER,
                    description VARCHAR2(100),
                    order_total NUMBER(8,2))
ORGANIZATION EXTERNAL
(
  TYPE oracle_hdfs
  DEFAULT DIRECTORY DEFAULT_DIR
  ACCESS PARAMETERS
  (
    com.oracle.bigdata.colmap: {"col": "item_cnt",
"field": "order_line_item_count"}
    com.oracle.bigdata.overflow: {"action": "TRUNCATE", "col": "DESCRIPTION"}
    com.oracle.bigdata.erroropt: [{"action": "replace", \
                                "value": "INVALID NUM", \
                                "col": ["CUST_NUM", "ORDER_NUM"]} , \
                                {"action": "reject", "col": "ORDER_TOTAL"}]
  )
  LOCATION ('hdfs:/usr/cust/summary/*');
```

The parameters make the following changes in the way that the ORACLE_HDFS access driver locates the data and handles error conditions:

- `com.oracle.bigdata.colmap`: Handles differences in column names. `ORDER_LINE_ITEM_COUNT` in the HDFS files matches the `ITEM_CNT` column in the external table.
- `com.oracle.bigdata.overflow`: Truncates string data. Values longer than 100 characters for the `DESCRIPTION` column are truncated.
- `com.oracle.bigdata.erroropt`: Replaces bad data. Errors in the data for `CUST_NUM` or `ORDER_NUM` set the value to `INVALID_NUM`.

3.4.2.2 Accessing Avro Container Files

The next example uses a SerDe to access Avro container files.

```
CREATE TABLE order (cust_num VARCHAR2(10),
                    order_num VARCHAR2(20),
                    order_date DATE,
                    item_cnt NUMBER,
                    description VARCHAR2(100),
                    order_total NUMBER(8,2))
ORGANIZATION EXTERNAL
(
  TYPE oracle_hdfs
  DEFAULT DIRECTORY DEFAULT_DIR
  ACCESS PARAMETERS (
    com.oracle.bigdata.rowformat: \
    SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
    com.oracle.bigdata.fileformat: \
    INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'\
```

```

OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
com.oracle.bigdata.colmap: { "col": "item_cnt", \
    "field": "order_line_item_count" }
com.oracle.bigdata.overflow: { "action": "TRUNCATE", \
    "col": "DESCRIPTION" }
)
LOCATION ('hdfs://usr/cust/summary/*');

```

The access parameters provide the following information to the ORACLE_HDFS access driver:

- `com.oracle.bigdata.rowformat`: Identifies the SerDe that the access driver needs to use to parse the records and fields. The files are not in delimited text format.
- `com.oracle.bigdata.fileformat`: Identifies the Java classes that can extract records and output them in the desired format.
- `com.oracle.bigdata.colmap`: Handles differences in column names. ORACLE_HDFS matches ORDER_LINE_ITEM_COUNT in the HDFS files with the ITEM_CNT column in the external table.
- `com.oracle.bigdata.overflow`: Truncates string data. Values longer than 100 characters for the DESCRIPTION column are truncated.

3.5 About the SQL CREATE TABLE Statement

The SQL CREATE TABLE statement has a clause specifically for creating external tables. The information that you provide in this clause enables the access driver to read data from an external source and prepare the data for the external table.

3.5.1 Basic Syntax

The following is the basic syntax of the CREATE TABLE statement for external tables:

```

CREATE TABLE table_name (column_name datatype,
                           column_name datatype[,...])
    ORGANIZATION EXTERNAL (external_table_clause);

```

You specify the column names and data types the same as for any other table. ORGANIZATION EXTERNAL identifies the table as an external table.

The *external_table_clause* identifies the access driver and provides the information that it needs to load the data. See [“About the External Table Clause”](#).

3.5.2 About the External Table Clause

CREATE TABLE ORGANIZATION EXTERNAL takes the *external_table_clause* as its argument. It has the following subclauses:

- [TYPE Clause](#)
- [DEFAULT DIRECTORY Clause](#)
- [LOCATION Clause](#)
- [REJECT LIMIT Clause](#)
- [ORACLE_HIVE Access Parameters](#)

See Also:*Oracle Database SQL Language Reference* for the `external_table_clause`

3.5.2.1 TYPE Clause

The `TYPE` clause identifies the access driver. The type of access driver determines how the other parts of the external table definition are interpreted.

Specify one of the following values for Oracle Big Data SQL:

- `ORACLE_HDFS`: Accesses files in an HDFS directory.
- `ORACLE_HIVE`: Accesses a Hive table.

Note:

The `ORACLE_DATAPUMP` and `ORACLE_LOADER` access drivers are not associated with Oracle Big Data SQL.

3.5.2.2 DEFAULT DIRECTORY Clause

The `DEFAULT DIRECTORY` clause identifies an Oracle Database directory object. The directory object identifies an operating system directory with files that the external table reads and writes.

`ORACLE_HDFS` and `ORACLE_HIVE` use the default directory solely to write log files on the Oracle Database system.

3.5.2.3 LOCATION Clause

The `LOCATION` clause identifies the data source.

3.5.2.4 ORACLE_HDFS LOCATION Clause

The `LOCATION` clause for `ORACLE_HDFS` contains a comma-separated list of file locations. The files must reside in the HDFS file system on the default cluster.

A location can be any of the following:

- A fully qualified HDFS name, such as `/user/hive/warehouse/hive_seed/hive_types`. `ORACLE_HDFS` uses all files in the directory.
- A fully qualified HDFS file name, such as `/user/hive/warehouse/hive_seed/hive_types/hive_types.csv`
- A URL for an HDFS file or a set of files, such as `hdfs:/user/hive/warehouse/hive_seed/hive_types/*`. Just a directory name is invalid.

The file names can contain any pattern-matching character described in [Table 3-2](#).

Table 3-2 *Pattern-Matching Characters*

Character	Description
?	Matches any one character
*	Matches zero or more characters

Table 3-2 (Cont.) Pattern-Matching Characters

Character	Description
[<i>abc</i>]	Matches one character in the set { <i>a</i> , <i>b</i> , <i>c</i> }
[<i>a-b</i>]	Matches one character in the range { <i>a...b</i> }. The character must be less than or equal to <i>b</i> .
[<i>^a</i>]	Matches one character that is not in the character set or range { <i>a</i> }. The carat (^) must immediately follow the left bracket, with no spaces.
\c	Removes any special meaning of <i>c</i> . The backslash is the escape character.
{ <i>ab\,cd</i> }	Matches a string from the set { <i>ab</i> , <i>cd</i> }. The escape character (\) removes the meaning of the comma as a path separator.
{ <i>ab\,c{de\,fh</i> }	Matches a string from the set { <i>ab</i> , <i>cde</i> , <i>cfh</i> }. The escape character (\) removes the meaning of the comma as a path separator.

3.5.2.5 ORACLE_HIVE LOCATION Clause

Do not specify the `LOCATION` clause for `ORACLE_HIVE`; it raises an error. The data is stored in Hive, and the access parameters and the metadata store provide the necessary information.

3.5.2.6 REJECT LIMIT Clause

Limits the number of conversion errors permitted during a query of the external table before Oracle Database stops the query and returns an error.

Any processing error that causes a row to be rejected counts against the limit. The reject limit applies individually to each parallel query (PQ) process. It is not the total of all rejected rows for all PQ processes.

3.5.2.7 ACCESS PARAMETERS Clause

The `ACCESS PARAMETERS` clause provides information that the access driver needs to load the data correctly into the external table. See [“CREATE TABLE ACCESS PARAMETERS Clause”](#).

3.6 About Data Type Conversions

When the access driver loads data into an external table, it verifies that the Hive data can be converted to the data type of the target column. If they are incompatible, then the access driver returns an error. Otherwise, it makes the appropriate data conversion.

Hive typically provides a table abstraction layer over data stored elsewhere, such as in HDFS files. Hive uses a serializer/deserializer (SerDe) to convert the data as needed from its stored format into a Hive data type. The access driver then converts the data from its Hive data type to an Oracle data type. For example, if a Hive table over a text file has a `BIGINT` column, then the SerDe converts the data from text to `BIGINT`. The access driver then converts the data from `BIGINT` (a Hive data type) to `NUMBER` (an Oracle data type).

Performance is better when one data type conversion is performed instead of two. The data types for the fields in the HDFS files should therefore indicate the data that is

actually stored on disk. For example, JSON is a clear text format, therefore all data in a JSON file is text. If the Hive type for a field is `DATE`, then the SerDe converts the data from string (in the data file) to a Hive date. Then the access driver converts the data from a Hive date to an Oracle date. However, if the Hive type for the field is string, then the SerDe does not perform a conversion, and the access driver converts the data from string to an Oracle date. Queries against the external table are faster in the second example, because the access driver performs the only data conversion.

[Table 3-3](#) identifies the data type conversions that `ORACLE_HIVE` can make when loading data into an external table.

Table 3-3 Supported Hive to Oracle Data Type Conversions

Hive Data Type	VARCHAR 2, CHAR, NCHAR2, NCHAR, CLOB	NUMBER, FLOAT, BINARY_NUMBER, BINARY_FLOAT	BLOB	RAW	DATE, TIMESTAMP, TIMESTAMP WITH TZ, TIMESTAMP WITH LOCAL TZ	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
INT SMALLINT TINYINT BIGINT	yes	yes	yes	yes	no	no
DOUBLE FLOAT	yes	yes	yes	yes	no	no
DECIMAL	yes	yes	no	no	no	no
BOOLEAN	yes ¹	yes	yes ²	yes	no	no
BINARY	yes	no	yes	yes	no	no
STRING	yes	yes	yes	yes	yes	yes
TIMESTAMP	yes	no	no	no	yes	no
STRUCT ARRAY UNIONTYPE MAP	yes	no	no	no	no	no

¹ FALSE maps to the string FALSE, and TRUE maps to the string TRUE.

² FALSE maps to 0, and TRUE maps to 1.

3.7 Querying External Tables

Users can query external tables using the SQL `SELECT` statement, the same as they query any other table.

3.7.1 Granting User Access

Users who query the data on a Hadoop cluster must have `READ` access in Oracle Database to the external table and to the database directory object that points to the cluster directory. See [“About the Cluster Directory”](#).

3.7.2 About Error Handling

By default, a query returns no data if an error occurs while the value of a column is calculated. Processing continues after most errors, particularly those thrown while the column values are calculated.

Use the [com.oracle.bigdata.erroropt](#) parameter to determine how errors are handled.

3.7.3 About the Log Files

You can use these access parameters to customize the log files:

- [com.oracle.bigdata.log.exec](#)
- [com.oracle.bigdata.log.qc](#)

3.8 About Oracle Big Data SQL on the Database Server (Oracle Exadata Machine or Other)

This section explains the changes that the Oracle Big Data SQL installation makes to the Oracle Database system (which may or may not be an Oracle Exadata Machine).

The section contains the following topics:

- [About the Common Directory](#)
- [Common Configuration Properties](#)
- [About the Cluster Directory](#)

3.8.1 About the Common Directory

The directory `common` directory contains configuration information that is common to all Hadoop clusters. This directory is located on the Oracle Database system under the Oracle home directory. The `oracle` file system user (or whichever user owns the Oracle Database instance) owns the common directory. A database directory named `ORACLE_BIGDATA_CONFIG` points to `common`.

3.8.2 Common Configuration Properties

The installation store these files in the `common` directory under `/home/oracle`:

- [bigdata.properties](#)
- [bigdata-log4j.properties](#)

The Oracle DBA can edit these configuration files as necessary.

3.8.2.1 bigdata.properties

The `bigdata.properties` file in the common directory contains property-value pairs that define the Java class paths and native library paths required for accessing data in HDFS.

These properties must be set:

- [bigdata.cluster.default](#)
- [java.classpath.hadoop](#)

- [java.classpath.hive](#)
- [java.classpath.oracle](#)

The following list describes all properties permitted in `bigdata.properties`.

bigdata.properties

bigdata.cluster.default	The name of the default Hadoop cluster. The access driver uses this name when the access parameters do not specify a cluster. Required. Changing the default cluster name might break external tables that were created previously without an explicit cluster name.
bigdata.cluster.list	A comma-separated list of Hadoop cluster names. Optional.
java.classpath.hadoop	The Hadoop class path. Required.
java.classpath.hive	The Hive class path. Required.
java.classpath.oracle	The path to the Oracle JXAD Java JAR file. Required.
java.classpath.user	The path to user JAR files. Optional.
java.libjvm.file	The full file path to the JVM shared library (such as <code>libjvm.so</code>). Required.
java.options	A comma-separated list of options to pass to the JVM. Optional. This example sets the maximum heap size to 2 GB, and verbose logging for Java Native Interface (JNI) calls: <code>Xmx2048m,-verbose=jni</code>
LD_LIBRARY_PATH	A colon separated (:) list of directory paths to search for the Hadoop native libraries. Recommended. If you set this option, then do not set <code>java.library</code> path in java.options .

[Example 3-1](#) shows a sample `bigdata.properties` file.

Example 3-1 Sample `bigdata.properties` File

```
# bigdata.properties
#
# Copyright (c) 2014, Oracle and/or its affiliates. All rights reserved.
#
#   NAME
#     bigdata.properties - Big Data Properties File
#
#   DESCRIPTION
#     Properties file containing parameters for allowing access to Big Data
```



```
#      Fixed value properties can be added here
#
java.libjvm.file=$ORACLE_HOME/jdk/jre/lib/amd64/server/libjvm.so
java.classpath.oracle=$ORACLE_HOME/hadoopcore/jlib/*:$ORACLE_HOME/hadoop/jlib/
hver-2/*:$ORACLE_HOME/dbjava/lib/*
java.classpath.hadoop=$HADOOP_HOME/*:$HADOOP_HOME/lib/*
java.classpath.hive=$HIVE_HOME/lib/*
LD_LIBRARY_PATH=$ORACLE_HOME/jdk/jre/lib
bigdata.cluster.default=hadoop_cl_1
```

3.8.2.2 bigdata-log4j.properties

The `bigdata-log4j.properties` file in the common directory defines the logging behavior of queries against external tables in the Java code. Any `log4j` properties are allowed in this file.

[Example 3-2](#) shows a sample `bigdata-log4j.properties` file with the relevant `log4j` properties.

Example 3-2 Sample bigdata-log4j.properties File

```
# bigdata-log4j.properties
#
# Copyright (c) 2014, Oracle and/or its affiliates. All rights reserved.
#
#      NAME
#      bigdata-log4j.properties - Big Data Logging Properties File
#
#      DESCRIPTION
#      Properties file containing logging parameters for Big Data
#      Fixed value properties can be added here

bigsql.rootlogger=INFO,console
log4j.rootlogger=DEBUG, file
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}: %m%n
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}: %m%n
log4j.logger.oracle.hadoop.sql=ALL, file

bigsql.log.dir=.
bigsql.log.file=bigsql.log
log4j.appender.file.File=$ORACLE_HOME/bigdatalogs/bigdata-log4j.log
```

See Also:

Apache Logging Services documentation at

<http://logging.apache.org/log4j/1.2/manual.html>

3.8.3 About the Cluster Directory

The cluster directory contains configuration information for a Hadoop cluster. Each cluster that Oracle Database will access using Oracle Big Data SQL has a cluster directory. This directory is located on the Oracle Database system under the common

directory. For example, a cluster named `bda1_cl_1` would have a directory by the same name (`bda1_cl_1`) in the common directory.

The cluster directory contains the client configuration files for accessing the cluster, such as the following:

- `core-site.xml`
- `hdfs-site.xml`
- `hive-site.xml`
- `mapred-site.xml` (optional)
- `log4j` property files (such as `hive-log4j.properties`)

A database directory object points to the cluster directory. Users who want to access the data in a cluster must have read access to the directory object.

3.8.4 About Permissions

- On the Oracle database server, the `oracle` user (or whatever user owns the Oracle Database installation directory) requires `READ/WRITE` access to the database directory that points to the log directory.
- There must also be a corresponding `oracle` user defined in the Hadoop cluster. The `oracle` user requires `READ` access to HDFS on all `DataNodes` in the cluster.

Copying Oracle Tables to Hadoop

This chapter describes how to use Copy to Hadoop to copy tables in an Oracle database to Hadoop. It contains the following sections:

- [What Is Copy to Hadoop?](#)
- [Getting Started Using Copy to Hadoop](#)
- [Installing Copy to Hadoop](#)
- [Generating the Data Pump Files](#)
- [Creating a Hive Table](#)
- [Example Using the Sample Schemas](#)

4.1 What Is Copy to Hadoop?

Oracle Big Data SQL includes the Oracle Copy to Hadoop utility. This utility makes it simple to identify and copy Oracle data to the Hadoop Distributed File System. It can be accessed either through a command-line interface or via Oracle SQL Developer. Data exported to the Hadoop cluster by Copy to Hadoop is stored in Oracle Data Pump format. This format optimizes queries thru Big Data SQL:

- The data is stored as Oracle data types – eliminating data type conversions
- The data is queried directly – without requiring the overhead associated with Java SerDes

After generating Data Pump format files from the tables and copying the files to HDFS, you can use Apache Hive to query the data. Hive can process the data locally without accessing Oracle Database. When the Oracle table changes, you can refresh the copy in Hadoop. Copy to Hadoop is primarily useful for Oracle tables that are relatively static, and thus do not require frequent refreshes.

Copy to Hadoop is licensed under Oracle Big Data SQL. You must have an Oracle Big Data SQL license in order to use utility

4.2 Getting Started Using Copy to Hadoop

Take the following steps to use Copy to Hadoop:

1. Ensure that your system meets the prerequisites, and that the required software is installed on both the Hadoop cluster (on Oracle Big Data Appliance or another Hadoop system) and on the Oracle Database server (Oracle Exadata Database Machine or other).

See [“Installing Copy to Hadoop”](#).

2. On the Oracle Database server, connect to Oracle Database and generate Data Pump format files containing the table data and metadata.

See [“Generating the Data Pump Files”](#).

3. Copy the files to HDFS on the Hadoop cluster.

See [“Copying the Files to HDFS”](#).

4. Connect to Apache Hive and create an external table from the files.

See [“Creating a Hive Table”](#).

5. Query this Hive table the same as you would any other Hive table.

4.3 Installing Copy to Hadoop

Where Oracle Big Data SQL is installed, Copy to Hadoop is available on the Oracle Database server that is connected to Hadoop cluster.

4.3.1 Prerequisites for Copy to Hadoop

For network connections, supported Oracle Database levels, and other requirements, see the general requirements for installing Oracle Big Data SQL on your platform: [Installing Oracle Big Data SQL](#).

4.3.2 Installing Copy to Hadoop in an Oracle Big Data Appliance/Oracle Exadata Database Machine Environment

If Oracle Big Data SQL is installed, Copy to Hadoop is available on the Oracle Exadata Database Machine connected to Oracle Big Data Appliance.

4.3.2.1 Installing Copy to Hadoop on Oracle Big Data Appliance

Copy to Hadoop is a component of Oracle Big Data SQL, which is an installation option on Oracle Big Data Appliance. You can enable Oracle Big Data SQL either during the initial software installation or at a later time using the standard methods for enabling and disabling services. See [“Performing the Installation”](#).

4.3.2.2 Installing Copy to Hadoop on Oracle Exadata Database Machine

Copy to Hadoop only requires a Hadoop client on Oracle Exadata Database Machine. It does not employ the additional software required by Oracle Big Data SQL.

If you plan to use Oracle Big Data SQL, then the Hadoop client is created automatically when you run the `bds-exa-install.sh` installation script. In this case, you do not need to take any additional steps. See [“Running the Post-Installation Script for Oracle Big Data SQL”](#).

If you do not plan to use Oracle Big Data SQL at this time, then you can install the Hadoop client manually instead of running the script.

4.3.3 Installing Copy to Hadoop on Other Systems

Where Oracle Big Data SQL is installed, Copy to Hadoop is available on the Oracle Database server that is connected to Hadoop cluster.

4.3.3.1 Installing Copy to Hadoop on a Hadoop Cluster (Other than an Oracle Big Data Database Appliance Cluster)

Copy to Hadoop is a component of Oracle Big Data SQL and requires some additional setups after the Oracle Big Data SQL installation. See the following MOS note for instructions — *Big Data SQL 3.0: Installing and Configuring Copy to Hadoop* (Doc ID 2115762.1). Contact Oracle Support if you have any questions.

4.3.3.2 Installing Copy to Hadoop on an Oracle Database Server (Other Than Oracle Exadata Database Machine)

In order to use Copy to Hadoop, additional configuration steps are required after the Oracle Big Data SQL installation. See the following MOS note for instructions — *Big Data SQL 3.0: Installing and Configuring Copy to Hadoop* (Doc ID 2115762.1). Contact Oracle Support if you have any questions.

4.4 Generating the Data Pump Files

The SQL `CREATE TABLE` statement has a clause specifically for creating external tables, in which you specify the `ORACLE_DATAPUMP` access driver. The information that you provide in this clause enables the access driver to generate a Data Pump format file that contains the data and metadata from the Oracle database table.

This section contains the following topics:

- [About Data Pump Format Files](#)
- [Identifying the Target Directory](#)
- [About the CREATE TABLE Syntax](#)
- [Copying the Files to HDFS](#)

4.4.1 About Data Pump Format Files

Data Pump files are typically used to move data and metadata from one database to another. Copy to Hadoop uses this file format to copy data from an Oracle database to HDFS.

To generate Data Pump format files, you create an external table from an existing Oracle table. An **external table** in Oracle Database is an object that identifies and describes the location of data outside of a database. External tables use **access drivers** to parse and format the data. For Copy to Hadoop, you use the `ORACLE_DATAPUMP` access driver. It copies the data and metadata from internal Oracle tables and populates the Data Pump format files of the external table.

4.4.2 Identifying the Target Directory

You must have read and write access to a database directory in Oracle Database. Only Oracle Database users with the `CREATE ANY DIRECTORY` system privilege can create directories.

This example creates a database directory named `EXPORTDIR` that points to the `/exportdir` directory on the Oracle Database server (Oracle Exadata Database Machine or other):

```
SQL> CREATE DIRECTORY exportdir AS '/exportdir';
```

4.4.3 About the CREATE TABLE Syntax

The following is the basic syntax of the `CREATE TABLE` statement for Data Pump format files:

```
CREATE TABLE table_name
  ORGANIZATION EXTERNAL (
    TYPE oracle_datapump
    DEFAULT DIRECTORY database_directory
    LOCATION ('filename1.dmp','filename2.dmp'...)
  ) PARALLEL n
  AS SELECT * FROM tablename;
```

DEFAULT DIRECTORY

Identifies the database directory that you created for this purpose. See [“Identifying the Target Directory”](#).

LOCATION

Lists the names of the Data Pump files to be created. The number of names should match the degree of parallelism (DOP) specified by the `PARALLEL` clause. Otherwise, the DOP drops to the number of files.

The number of files and the degree of parallelism affect the performance of Oracle Database when generating the Data Pump format files. They do not affect querying performance in Hive.

PARALLEL

Sets the degree of parallelism (DOP). Use the maximum number that your Oracle DBA permits you to use. By default the DOP is 1, which is serial processing. Larger numbers enable parallel processing.

AS SELECT

Use the full SQL `SELECT` syntax for this clause. It is not restricted. The *tablename* identifies the Oracle table to be copied to HDFS.

See Also:

For descriptions of these parameters:

- *Oracle Database SQL Language Reference*
 - *Oracle Database Utilities*
-
-

4.4.4 Copying the Files to HDFS

The Oracle Big Data SQL installation installs Hadoop client files on the Oracle Database server (Oracle Exadata Database Machine or other). The Hadoop client installation enables you to use Hadoop commands to copy the Data Pump files to HDFS. You must have write privileges on the HDFS directory.

To copy the `dmp` files into HDFS, use the `hadoop fs -put` command. This example copies the files into the HDFS `customers` directory owned by the `oracle` user:

```
$ hadoop fs -put customers*.dmp /user/oracle/customers
```

4.5 Creating a Hive Table

To provide access to the data in the Data Pump files, you create a Hive external table over the Data Pump files. Copy to Hadoop provides SerDes that enable Hive to read the files. These SerDes are read only, so you cannot use them to write to the files.

See Also:

Apache Hive Language Manual DDL at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable>

4.5.1 About Hive External Tables

For external tables, Hive loads the table metadata into its metastore. The data remains in its original location, which you identify in the `LOCATION` clause. If you drop an external table using a HiveQL `DROP TABLE` statement, then only the metadata is discarded, while the external data remains unchanged. In this respect, Hive handles external tables in fundamentally the same way as Oracle Database.

External tables support data sources that are shared by multiple programs. In this case, you use Oracle Database to update the data and then generate a new file. You can overwrite the old HDFS files with the updated files while leaving the Hive metadata intact.

The following is the basic syntax of a Hive `CREATE TABLE` statement for creating a Hive external table for use with a Data Pump format file:

```
CREATE EXTERNAL TABLE tablename
ROW FORMAT
  SERDE 'oracle.hadoop.hive.datapump.DPSerDe'
STORED AS
  INPUTFORMAT 'oracle.hadoop.hive.datapump.DPInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 'hdfs_directory'
```

4.5.2 About Column Mappings

The Hive table columns automatically have the same names as the Oracle columns, which are provided by the metadata stored in the Data Pump files. Any user-specified column definitions are ignored.

4.5.3 About Data Type Conversions

Copy to Hadoop automatically converts the data in an Oracle table to an appropriate Hive data type. [Table 4-1](#) shows the default mappings between Oracle and Hive data types.

Table 4-1 Oracle to Hive Data Type Conversions

Table 4-1 (Cont.) Oracle to Hive Data Type Conversions

Oracle Data Type	Hive Data Type
NUMBER	INT when the scale is 0 and the precision is less than 10 BIGINT when the scale is 0 and the precision is less than 19 DECIMAL when the scale is greater than 0 or the precision is greater than 19
CLOB NCLOB	STRING
BINARY_DOUBLE	DOUBLE
BINARY_FLOAT	FLOAT
BLOB	BINARY
CHAR NCHAR	CHAR
VARCHAR2 NVARCHAR2	VARCHAR
ROWID UROWID	BINARY
DATE	TIMESTAMP
TIMESTAMP	TIMESTAMP
TIMESTAMPtz ¹ TIMESTAMPtz	Unsupported
RAW	BINARY

¹ To copy `TIMESTAMPtz` and `TIMESTAMPtz` data to Hive, cast the columns to `TIMESTAMP` when exporting them to the Data Pump files. Hive does not have a data type that supports time zones or time offsets.

4.6 Example Using the Sample Schemas

This example shows all steps in the process of creating a Hive table from an Oracle table using Copy to Hadoop.

4.6.1 About the Sample Data

The Oracle tables are from the Sales History (SH) sample schema. The `CUSTOMERS` table provides extensive information about individual customers, including names, addresses, telephone numbers, birth dates, and credit limits. The `COUNTRIES` table provides a list of countries, and identifies regions and subregions.

This query shows a small selection of data in the `CUSTOMERS` table:

```
SELECT cust_first_name first_name,
       cust_last_name last_name,
       cust_gender gender,
       cust_year_of_birth birth
```



```
FROM customers
ORDER BY cust_city, last_name
FETCH FIRST 10 ROWS ONLY;
```

The query returns the following rows:

FIRST_NAME	LAST_NAME	GENDER	BIRTH
Lise	Abbey	F	1963
Lotus	Alden	M	1958
Emmanuel	Aubrey	M	1933
Phil	Ball	M	1956
Valentina	Bardwell	F	1965
Lolita	Barkley	F	1966
Heloise	Barnes	M	1980
Royden	Barrett	M	1937
Gilbert	Braun	M	1984
Portia	Capp	F	1948

To reproduce this example, install the sample schemas in Oracle Database and connect as the SH user.

See Also:

Oracle Database Sample Schemas for descriptions of the tables and installation instructions for the schemas.

4.6.2 Creating the EXPDIR Database Directory

These SQL statements create a local database directory named EXPDIR and grant access to the SH user:

```
SQL> CREATE DIRECTORY expdir AS '/expdir';
Directory created.
SQL> GRANT READ, WRITE ON DIRECTORY expdir TO SH;
Grant succeeded.
```

4.6.3 Creating Data Pump Format Files for Customer Data

The following examples show how to create the Data Pump files and check their contents.

Copy to Hadoop supports only the syntax shown in the examples. Data pump files created with the Export utility or Oracle Data Pump are not compatible.

4.6.3.1 CREATE TABLE Example With a Simple SELECT Statement

This example shows a very simple SQL command for creating a Data Pump format file from the CUSTOMERS table. It selects the entire table and generates a single output file named `customers.dmp` in the local `/expdir` directory.

```
CREATE TABLE export_customers
  ORGANIZATION EXTERNAL
  (
    TYPE oracle_datapump
    DEFAULT DIRECTORY expdir
    LOCATION('customers.dmp')
  )
AS SELECT * FROM customers;
```

4.6.3.2 CREATE TABLE Example With a More Complex SQL SELECT Statement

The next example shows more complexity in the syntax. It joins the CUSTOMERS and COUNTRIES tables on the COUNTRY_ID columns to provide the country names. It also limits the rows to customers in the Americas. The command generates two output files in parallel, named `americas1.dmp` and `americas2.dmp`, in the local `/expdir` directory.

```
CREATE TABLE export_americas
  ORGANIZATION EXTERNAL
  (
    TYPE oracle_datapump
    DEFAULT DIRECTORY expdir
    LOCATION('americas1.dmp', 'americas2.dmp')
  )
  PARALLEL 2
AS SELECT a.cust_first_name first_name,
       a.cust_last_name last_name,
       a.cust_gender gender,
       a.cust_year_of_birth birth,
       a.cust_email email,
       a.cust_postal_code postal_code,
       b.country_name country
FROM customers a,
     countries b
WHERE a.country_id=b.country_id AND
      b.country_region='Americas'
ORDER BY a.country_id, a.cust_postal_code;
```

4.6.4 Verifying the Contents of the Data Files

You can check the content of the output data files before copying them to Hadoop. The previous CREATE TABLE statement created an external table named EXPORT_AMERICAS, which you can describe and query the same as any other table.

The DESCRIBE statement shows the selection of columns and the modified names:

```
SQL> DESCRIBE export_americas;
Name                               Null?    Type
-----
FIRST_NAME                         NOT NULL VARCHAR2(20)
LAST_NAME                         NOT NULL VARCHAR2(40)
GENDER                            NOT NULL CHAR(1)
BIRTH                             NOT NULL NUMBER(4)
EMAIL                             VARCHAR2(50)
POSTAL_CODE                       NOT NULL VARCHAR2(10)
COUNTRY                           NOT NULL VARCHAR2(40)
```

A SELECT statement like the following shows a sample of the data:

```
SELECT first_name, last_name, gender, birth, country
FROM export_americas
WHERE birth > 1985
ORDER BY last_name
FETCH FIRST 5 ROWS ONLY;
```

FIRST_NAME	LAST_NAME	GENDER	BIRTH	COUNTRY
Opal	Aaron	M	1990	United States of America
KaKit	Abeles	M	1986	United States of America
Mitchel	Alambarati	M	1987	Canada

Jade	Anderson	M	1986 United States of America
Roderica	Austin	M	1986 United States of America

4.6.5 Copying the Files into Hadoop

The following commands list the files in the local `expdir` directory, create a Hadoop subdirectory named `customers`, and copy the files to it. The user is connected to the Hadoop cluster (Oracle Big Data Appliance or other) as the `oracle` user.

```
$ cd /expdir
$ ls americas*.dmp
americas1.dmp  americas2.dmp
$ hadoop fs -mkdir customers
$ hadoop fs -put *.dmp customers
$ hadoop fs -ls customers
Found 2 items
-rw-r--r--  1 oracle oracle    798720 2014-10-13 17:04 customers/americas1.dmp
-rw-r--r--  1 oracle oracle    954368 2014-10-13 17:04 customers/americas2.dmp
```

4.6.6 Creating a Hive External Table

This HiveQL statement creates an external table using the Copy to Hadoop SerDes. The `LOCATION` clause identifies the full path to the Hadoop directory containing the Data Pump files:

```
CREATE EXTERNAL TABLE customers
  ROW FORMAT SERDE 'oracle.hadoop.hive.datapump.DPSerDe'
  STORED AS
    INPUTFORMAT 'oracle.hadoop.hive.datapump.DPInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
  LOCATION '/user/oracle/customers';
```

The `DESCRIBE` command shows the columns of the `CUSTOMERS` external table.

```
hive> DESCRIBE customers;
OK
first_name      varchar(20)      from deserializer
last_name       varchar(40)      from deserializer
gender          char(1)          from deserializer
birth           int              from deserializer
email           varchar(50)      from deserializer
postal_code     varchar(10)      from deserializer
country         varchar(40)      from deserializer
```

4.6.7 Querying the Data in Hive

The following HiveQL `SELECT` statement shows the same data as the SQL `SELECT` statement from Oracle Database shown in [“Verifying the Contents of the Data Files”](#). The two queries access copies of the same Data Pump files.

```
SELECT first_name, last_name, gender, birth, country
  FROM customers
 WHERE birth > 1985
 ORDER BY last_name LIMIT 5;
```

```
Total MapReduce jobs = 1
Launching Job 1 out of 1
```

```
  .
  .
  .
OK
```

Opal	Aaron	M	1990	United States of America
KaKit	Abeles	M	1986	United States of America
Mitchel	Alambarati	M	1987	Canada
Jade	Anderson	M	1986	United States of America
Roderica	Austin	M	1986	United States of America

Oracle Big Data SQL Reference

This chapter contains reference information for Oracle Big Data SQL:

- [DBMS_HADOOP PL/SQL Package](#)
- [CREATE TABLE ACCESS PARAMETERS Clause](#)
- [Static Data Dictionary Views for Hive](#)

5.1.1 DBMS_HADOOP PL/SQL Package

The DBMS_HADOOP package contains a function to generate the CREATE EXTERNAL TABLE DDL for a Hive table:

- [CREATE_EXTDDL_FOR_HIVE](#)

5.1.1.1 CREATE_EXTDDL_FOR_HIVE

This function returns a SQL CREATE TABLE ORGANIZATION EXTERNAL statement for a Hive table. It uses the ORACLE_HIVE access driver.

Syntax

```
DBMS_HADOOP.CREATE_EXTDDL_FOR_HIVE (
  cluster_id      IN   VARCHAR2,
  db_name         IN   VARCHAR2  := NULL,
  hive_table_name IN   VARCHAR2,
  hive_partition  IN   BOOLEAN,
  table_name      IN   VARCHAR2  := NULL,
  perform_ddl     IN   BOOLEAN   DEFAULT FALSE,
  text_of_ddl     OUT  VARCHAR2
);
```

Parameters

Table 5-1 CREATE_EXTDDL_FOR_HIVE Function Parameters

Parameter	Description
cluster_id	Hadoop cluster where the Hive metastore is located
db_name	Name of the Hive database
hive_table_name	Name of the Hive table
hive_partition	Whether the table is partitioned (TRUE) or not (FALSE)

Table 5-1 (Cont.) CREATE_EXTDDL_FOR_HIVE Function Parameters

Parameter	Description
table_name	Name of the Oracle external table to be created. It cannot already exist.
perform_ddl	Whether to execute the generated CREATE TABLE statement (TRUE) or just return the text of the command (FALSE). Do not execute the command automatically if you want to review or modify it.
text_of_ddl	The generated CREATE TABLE ORGANIZATION EXTERNAL statement.

Usage Notes

The Oracle Database system must be configured for Oracle Big Data SQL. See [“About Oracle Big Data SQL on the Database Server \(Oracle Exadata Machine or Other\)”](#).

The data type conversions are based on the default mappings between Hive data types and Oracle data types. See [“About Data Type Conversions”](#).

5.1.1.1.1 Example

The following query returns the CREATE EXTERNAL TABLE DDL for my_hive_table from the default Hive database. The connection to Hive is established using the configuration files in the ORACLE_BIGDATA_CONFIG directory, which identify the location of the HADOOP1 cluster.

```
DECLARE
    DDLtxt VARCHAR2(4000);
BEGIN
    dbms_hadoop.create_extddl_for_hive(
        CLUSTER_ID=>'hadoop1',
        DB_NAME=>'default',
        HIVE_TABLE_NAME=>'my_hive_table',
        HIVE_PARTITION=>FALSE,
        TABLE_NAME=>'my_xt_oracle',
        PERFORM_DDL=>FALSE,
        TEXT_OF_DDL=>DDLtxt
    );
    dbms_output.put_line(DDLtxt);
END;
/
```

The query returns the text of the following SQL command:

```
CREATE TABLE my_xt_oracle
(
    c0 VARCHAR2(4000),
    c1 VARCHAR2(4000),
    c2 VARCHAR2(4000),
    c3 VARCHAR2(4000))
ORGANIZATION EXTERNAL
    (TYPE ORACLE_HIVE
     DEFAULT DIRECTORY DEFAULT_DIR
     ACCESS PARAMETERS (
```

```

        com.oracle.bigdata.cluster=hadoop1
        com.oracle.bigdata.tablename=default.my_hive_table
    )
PARALLEL 2 REJECT LIMIT UNLIMITED

```

5.1.2 CREATE TABLE ACCESS PARAMETERS Clause

This section describes the properties that you use when creating an external table that uses the ORACLE_HDFS or ORACLE_HIVE access drivers. In a CREATE TABLE ORGANIZATION EXTERNAL statement, specify the parameters in the opaque_format_spec clause of ACCESS PARAMETERS.

This section contains the following topics:

- [Syntax Rules for Specifying Properties](#)
- [ORACLE_HDFS Access Parameters](#)
- [ORACLE_HIVE Access Parameters](#)
- Alphabetical list of properties

5.1.2.1 Syntax Rules for Specifying Properties

The properties are set using keyword-value pairs in the SQL CREATE TABLE ACCESS PARAMETERS clause and in the configuration files. The syntax must obey these rules:

- The format of each keyword-value pair is a *keyword*, a colon or equal sign, and a *value*. The following are valid keyword-value pairs:

```

keyword=value
keyword:value

```

The value is everything from the first non-whitespace character after the separator to the end of the line. Whitespace between the separator and the value is ignored. Trailing whitespace for the value is retained.

- A property definition can be on one line or multiple lines.
- A line terminator is a line feed, a carriage return, or a carriage return followed by line feeds.
- When a property definition spans multiple lines, then precede the line terminators with a backslash (escape character), except on the last line. In this example, the value of the Keyword1 property is Value part 1 Value part 2 Value part 3.

```

Keyword1= Value part 1 \
          Value part 2 \
          Value part 3

```

- You can create a *logical line* by stripping each physical line of leading whitespace and concatenating the lines. The parser extracts the property names and values from the logical line.
- You can embed special characters in a property name or property value by preceding a character with a backslash (escape character), indicating the substitution. [Table 5-2](#) describes the special characters.

Table 5-2 Special Characters in Properties

Escape Sequence	Character
\b	Backspace (\u0008)
\t	Horizontal tab (\u0009)
\n	Line feed (\u000a)
\f	Form feed (\u000c)
\r	Carriage return (\u000d)
\"	Double quote (\u0022)
\'	Single quote (\u0027)
\\	Backslash (\u005c) When multiple backslashes are at the end of the line, the parser continues the value to the next line only for an odd number of backslashes.
\uxxxx	2-byte, big-endian, Unicode code point. When a character requires two code points (4 bytes), the parser expects \u for the second code point.

5.1.2.2 ORACLE_HDFS Access Parameters

The access parameters for the `ORACLE_HDFS` access driver provide the metadata needed to locate the data in HDFS and generate a Hive table over it.

5.1.2.2.1 Default Parameter Settings for ORACLE_HDFS

If you omit all access parameters from the `CREATE TABLE` statement, then `ORACLE_HDFS` uses the following default values:

```
com.oracle.bigdata.rowformat=DELIMITED
com.oracle.bigdata.fileformat=TEXTFILE
com.oracle.bigdata.overflow={"action":"truncate"}
com.oracle.bigdata.erroropt={"action":"setnull"}
```

5.1.2.2.2 Optional Parameter Settings for ORACLE_HDFS

`ORACLE_HDFS` supports the following optional `com.oracle.bigdata` parameters, which you can specify in the `opaque_format_spec` clause:

- `com.oracle.bigdata.colmap`
- `com.oracle.bigdata.erroropt`
- `com.oracle.bigdata.fields`
- `com.oracle.bigdata.fileformat`
- `com.oracle.bigdata.log.exec`
- `com.oracle.bigdata.log.qc`

- [com.oracle.bigdata.overflow](#)
- [com.oracle.bigdata.rowformat](#)

Example 5-1 shows a CREATE TABLE statement in which multiple access parameters are set.

Example 5-1 Setting Multiple Access Parameters for ORACLE_HDFS

```
CREATE TABLE ORDER (CUST_NUM VARCHAR2(10),
                     ORDER_NUM VARCHAR2(20),
                     ORDER_DATE DATE,
                     ITEM_CNT NUMBER,
                     DESCRIPTION VARCHAR2(100),
                     ORDER_TOTAL (NUMBER8,2)) ORGANIZATION EXTERNAL
    (TYPE ORACLE_HDFS
    ACCESS PARAMETERS (
        com.oracle.bigdata.fields: (CUST_NUM,          \
                                     ORDER_NUM,         \
                                     ORDER_DATE,        \
                                     ORDER_LINE_ITEM_COUNT, \
                                     DESCRIPTION,       \
                                     ORDER_TOTAL)
        com.oracle.bigdata.colMap:      {"col": "item_cnt", \
                                         "field": "order_line_item_count"}
        com.oracle.bigdata.overflow:   {"action": "TRUNCATE", \
                                         "col": "DESCRIPTION"}
        com.oracle.bigdata.errorOpt:   [{"action": "replace", \
                                         "value": "INVALID NUM", \
                                         "col": ["CUST_NUM", "ORDER_NUM"]} , \
                                         {"action": "reject", \
                                         "col": "ORDER_TOTAL"}]
    )
    LOCATION ("hdfs:/usr/cust/summary/*");
```

5.1.2.3 ORACLE_HIVE Access Parameters

ORACLE_HIVE retrieves metadata about external data sources from the Hive catalog. The default mapping of Hive data to columns in the external table are usually appropriate. However, some circumstances require special parameter settings, or you might want to override the default values for reasons of your own.

5.1.2.3.1 Default Parameter Settings for ORACLE_HIVE

If you omit all access parameters from the CREATE TABLE statement, then ORACLE_HIVE uses the following default values:

```
com.oracle.bigdata.tablename=name of external table
com.oracle.bigdata.overflow={"action": "truncate"}
com.oracle.bigdata.erroropt={"action": "setnull"}
```

5.1.2.3.2 Optional Parameter Values for ORACLE_HIVE

ORACLE_HIVE supports the following optional com.oracle.bigdata parameters, which you can specify in the opaque_format_spec clause:

- [com.oracle.bigdata.colmap](#)
- [com.oracle.bigdata.erroropt](#)
- [com.oracle.bigdata.log.exec](#)

- [com.oracle.bigdata.log.qc](#)
- [com.oracle.bigdata.overflow](#)
- [com.oracle.bigdata.tablename](#)

Example 5-2 shows a CREATE TABLE statement in which multiple access parameters are set.

Example 5-2 Setting Multiple Access Parameters for ORACLE_HIVE

```
CREATE TABLE ORDER (cust_num VARCHAR2(10),
                     order_num VARCHAR2(20),
                     order_date DATE,
                     item_cnt NUMBER,
                     description VARCHAR2(100),
                     order_total (NUMBER8,2)) ORGANIZATION EXTERNAL
(TYPE oracle_hive
 ACCESS PARAMETERS (
   com.oracle.bigdata.tableName: order_db.order_summary
   com.oracle.bigdata.colMap:    {"col":"ITEM_CNT", \
                                   "field":"order_line_item_count"}
   com.oracle.bigdata.overflow: {"action":"ERROR", \
                                   "col":"DESCRIPTION"}
   com.oracle.bigdata.errorOpt: [{"action":"replace", \
                                   "value":"INV_NUM" , \
                                   "col":["CUST_NUM","ORDER_NUM"]} ,\
                                   {"action":"reject", \
                                   "col":"ORDER_TOTAL"}]
 ));
```

5.1.2.4 com.oracle.bigdata.colmap

Maps a column in the source data to a column in the Oracle external table. Use this property when the source field names exceed the maximum length of Oracle column names, or when you want to use different column names in the external table.

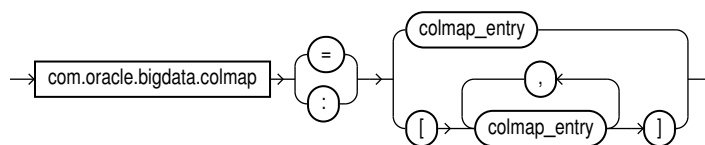
Default Value

A column in the external table with the same name as the Hive column

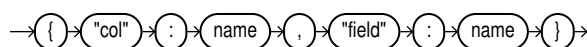
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

colmap ::=



colmap_entry ::=



Semantics

"col":name

`"col"`: The keyword must be lowercase and enclosed in quotation marks.

name: The name of a column in the Oracle external table. It is case sensitive and must be enclosed in quotation marks.

`"field":name`

`"field"`: The keyword must be lowercase and enclosed in quotation marks.

name: The name of a field in the data source. It is not case sensitive, but it must be enclosed in quotation marks. See [“Syntax Rules for Specifying Properties”](#).

Example

This example maps a Hive column named `ORDER_LINE_ITEM_COUNT` to an Oracle column named `ITEM_CNT`:

```
com.oracle.bigdata.colMap={"col":"ITEM_CNT", \
                           "field":"order_line_item_count"}
```

5.1.2.5 com.oracle.bigdata.datamode

Specifies the method that SmartScan uses to scan a Hadoop data source. The method can make a significant difference in performance.

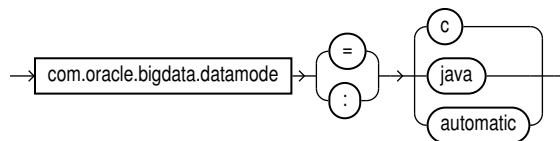
Default Value

`automatic`

Syntax

A JSON document with the keyword-value pairs shown in the following diagram:

`datamode ::=`



Semantics

`automatic`

Automatically selects the appropriate mode, based on the metadata. It selects `c` mode if possible, or `java` mode if the data contains formats that are not supported by `c` mode.

`c`

Uses Java to read the file buffers, but C code to process the data and convert it to Oracle format. Specify this mode for delimited data.

If the data contains formats that the C code does not support, then it returns an error.

`java`

Uses the Java SerDes and InputFormats to process the data and convert it to Oracle format. Specify this mode for Parquet, RCFile, and other data formats that require a SerDe.

5.1.2.6 com.oracle.bigdata.erroropt

Describes how to handle errors that occur while the value of a column is calculated.

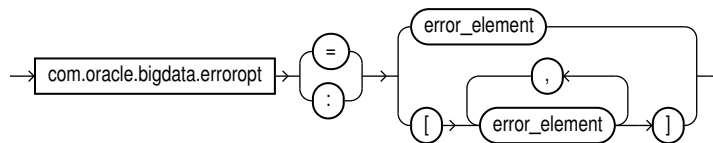
Default Value

```
{"action": "setnull"}
```

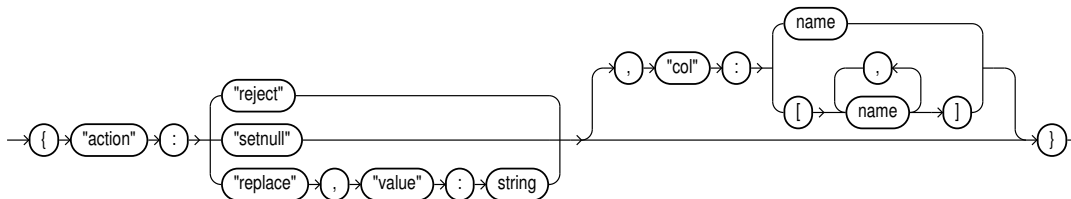
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

erroropt ::=



error_element ::=



Semantics

The "action", "reject", "setnull", "replace", "value", and "col" keywords must be lowercase and enclosed in quotation marks. See [“Syntax Rules for Specifying Properties”](#).

"action":*value*

value: One of these keywords:

- "reject": Does not load any rows.
- "setnull": Sets the column to NULL.
- "replace": Sets the column to the specified value.

"value":*string*

string: Replaces a bad value in the external table. It must be enclosed in quotation marks.

"col":*name*

name: Identifies a column in an external table. The column name is case sensitive, must be enclosed in quotation marks, and can be listed only once.

Example

This example sets the value of the CUST_NUM or ORDER_NUM columns to INVALID if the Hive value causes an error. For any other columns, an error just causes the Hive value to be rejected.

```
com.oracle.bigdata.errorOpt: { "action": "replace", \
                               "value": "INVALID", \
                               "col": [ "CUST_NUM", "ORDER_NUM" ] }
```

5.1.2.7 com.oracle.bigdata.fields

Lists the field names and data types of the data source.

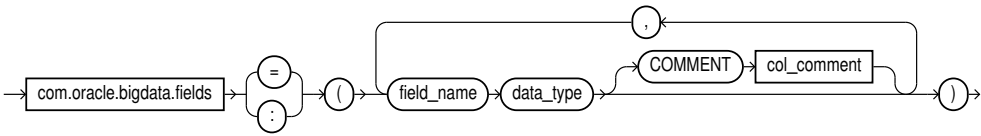
Default Value

Not defined

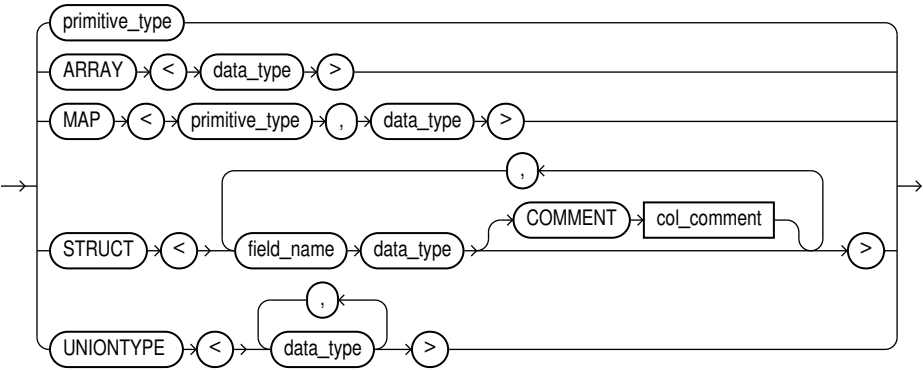
Syntax

A JSON document with the keyword-value pairs is shown in the following diagram:

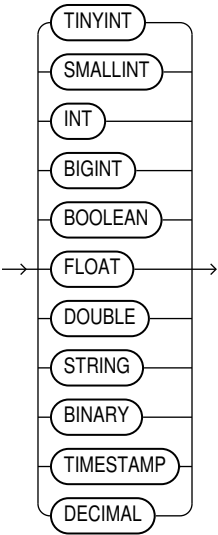
fields ::=



data_type ::=



primitive_type ::=



Semantics

The syntax is the same as a field list for a Hive table. If you split the field list across multiple lines, you must use a backslash to escape the new line characters.

field_name

The name of the Hive field. Use only alphanumeric characters and underscores (_). The maximum length is 128 characters. Field names are case-insensitive.

data_type

The data type of the Hive field. Optional; the default is `STRING`. The character set must be UTF8.

The data type can be complex or primitive:

Hive Complex Data Types

- `ARRAY`: Indexable list
- `MAP`: Key-value tuples
- `STRUCT`: List of elements
- `UNIONTYPE`: Multiple data types

Hive Primitive Data Types

- `INT`: 4 byte integer
- `BIGINT`: 8 byte integer
- `SMALLINT`: 2 byte integer
- `TINYINT`: 1 byte integer
- `BOOLEAN`: `TRUE` or `FALSE`
- `FLOAT`: single precision
- `DOUBLE`: double precision
- `STRING`: character sequence

See Also:

"Data Types" in the *Apache Hive Language Manual* at

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types>

`COMMENT col_comment`

A string literal enclosed in single quotation marks, which is stored as metadata for the Hive table (comment property of `TBLPROPERTIES`).

5.1.2.8 com.oracle.bigdata.fileformat

Describes the row format of the data source, based on the `ROW FORMAT` clause for a Hive table generated by `ORACLE_HDFS`.

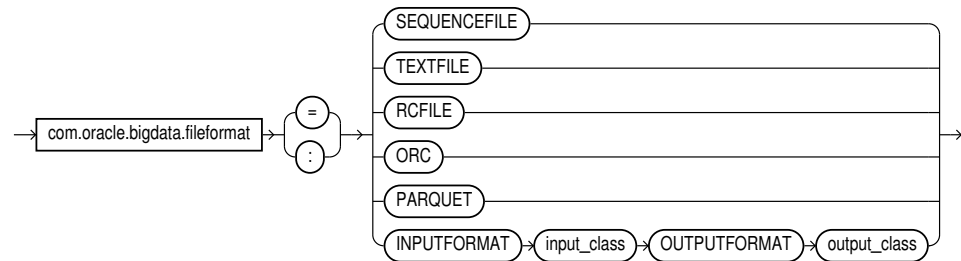
Default Value

TEXTFILE

Syntax

A JSON document with the keyword-value pairs is shown in the following diagram.

fileformat ::=

**Semantics**

ORC

Optimized row columnar file format

PARQUET

Column-oriented, binary file format

RCFILE

Record columnar file format

SEQUENCEFILE

Compressed file format

TEXTFILE

Plain text file format

INPUTFORMAT

Identifies a Java class that can extract records from the data file.

OUTPUTFORMAT

Identifies a Java class that can format the output records in the desired format

5.1.2.9 com.oracle.bigdata.log.exec

Specifies how the access driver generates log files generated by the C code for a query, when it is running as parallel processes on CDH.

The access driver does not create or write log files when executing on a Hadoop cluster node; the parallel query processes write them. The log files from the Java code are controlled by `log4j` properties, which are specified in the configuration file or the access parameters. See [“bigdata-log4j.properties”](#).

Default Value

Not defined (no logging)

Syntax

[directory_object:]file_name_template

Semantics

directory_object

The Oracle directory object for the HDFS path on the Hadoop cluster where the log file is created.

file_name_template

A string used to generate file names. [Table 5-2](#) describes the optional variables that you can use in the template.

Table 5-3 Variables for *com.oracle.bigdata.log.exec*

Variable	Value
%p	Operating system process identifier (PID)
%a	A number that uniquely identifies the process.
%%	A percent sign (%)

Example

The following example generates log file names that include the PID and a unique number, such as `xtlogp_hive14_3413_57`:

```
com.oracle.bigdata.log.exec= xtlogp_hive14_%p_%a
```

5.1.2.10 *com.oracle.bigdata.log.qc*

Specifies how the access driver generates log files for a query.

Default Value

Not defined (no logging)

Syntax

[directory_object:]file_name_template

Semantics

directory_object

Name of an Oracle directory object that points to the path where the log files are written. If this value is omitted, then the logs are written to the default directory for the external table.

file_name_template

A string used to generate file names. [Table 5-4](#) describes the optional variables that you can use in the string.

Table 5-4 Variables for *com.oracle.bigdata.log.qc*

100

[illegible]

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1



1000

- **error:** Throws an error. The [com.oracle.bigdata.erroropt](#) property controls the result of the error.

"col":*name*

name: Identifies a column in the external table. The name is case sensitive and must be enclosed in quotation marks.

Example

This example truncates the source data for the `DESCRIPTION` column, if it exceeds the column width:

```
com.oracle.bigdata.overflow={ "action":"truncate", \
                             "col":"DESCRIPTION" }
```

5.1.2.12 com.oracle.bigdata.rowformat

Provides the information the access driver needs to extract fields from the records in a file.

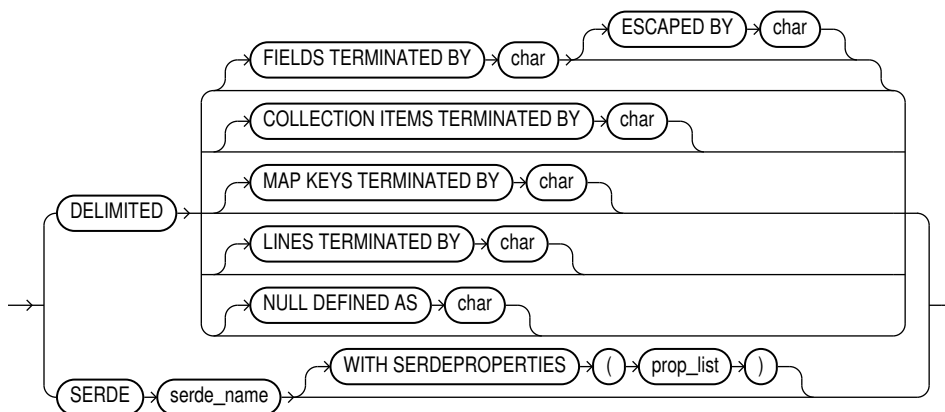
Default Value

DELIMITED

Syntax

A JSON document with the keyword-value pairs is shown in the following diagram.

rowformat ::=



Semantics

DELIMITED

Describes the characters used to delimit the fields in a record:

- **FIELDS TERMINATED BY:** The character that delimits every field in the record. The optional **ESCAPED BY** character precedes the delimit character when it appears within a field value.
- **COLLECTION ITEMS TERMINATED BY:** The character that marks the end of an array element.
- **MAP KEYS TERMINATED BY:** The character that marks the end of an entry in a MAP field.

- `LINES TERMINATED BY`: The character that marks the end of a record.
- `NULL DEFINED AS`: The character that indicates a null value.

SERDE

Identifies a SerDe that can parse the data and any properties of the SerDe that the access driver might need.

Example

This example specifies a SerDe for an Avro container file:

```
com.oracle.bigdata.rowformat:
  SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
```

The next example specifies a SerDe for a file containing regular expressions:

```
com.oracle.bigdata.rowformat=\
  SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe' \
  WITH SERDEPROPERTIES \
    ("input.regex" = "(\\\\d{6}) (\\\\d{5}) (.{29}) .*")
```

5.1.2.13 com.oracle.bigdata.tablename

Identifies the Hive table that contains the source data.

Default Value

DEFAULT.*external_table_name*

Syntax

[hive_database_name.]table_name

Semantics

The maximum length of *hive_database_name* and *table_name* is 128 UTF-8 characters (512 bytes).

hive_database_name

The Hive database where the source data resides. DEFAULT is the name of the initial Hive database.

table_name

The Hive table with the data. If you omit *table_name*, then ORACLE_HIVE searches for a Hive table with the same name as the external table. Table names are case-insensitive.

Example

This setting indicates that the source data is in a table named ORDER_SUMMARY in the Hive ORDER_DB database:

```
com.oracle.bigdata.tablename ORDER_DB.ORDER_SUMMARY
```

5.1.3 Static Data Dictionary Views for Hive

The Oracle Database catalog contains several static data dictionary views for Hive tables. You can query these data dictionary views to discover information about the Hive tables that you can access.

For you to access any Hive databases from Oracle Database, you must have read privileges on the `ORACLE_BIGDATA_CONFIG` directory object.

- [ALL_HIVE_DATABASES](#)
- [ALL_HIVE_TABLES](#)
- [ALL_HIVE_COLUMNS](#)
- [DBA_HIVE_DATABASES](#)
- [DBA_HIVE_TABLES](#)
- [DBA_HIVE_COLUMNS](#)
- [USER_HIVE_DATABASES](#)
- [USER_HIVE_TABLES](#)
- [USER_HIVE_COLUMNS](#)

5.1.3.1 ALL_HIVE_DATABASES

`ALL_HIVE_DATABASES` describes all databases in the Hive metastore accessible to the current user.

Related Views

- `DBA_HIVE_DATABASES` describes all the databases in the Hive metastore.
- `USER_HIVE_DATABASES` describes the databases in the Hive metastore owned by the current user.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2(4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2(4000)	NOT NULL	Hive database name
DESCRIPTION	VARCHAR2(4000)		Hive database description
DB_LOCATION	VARCHAR2(4000)	NOT NULL	
HIVE_URI	VARCHAR2(4000)		Hive database URI

See Also:

- [“DBA_HIVE_DATABASES”](#)
- [“USER_HIVE_DATABASES”](#)

5.1.3.2 ALL_HIVE_TABLES

ALL_HIVE_TABLES describes all tables in the Hive metastore accessible to the current user.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See [“About the Common Directory”](#).

Related Views

- DBA_HIVE_TABLES describes all tables in the Hive metastore.
- USER_HIVE_TABLES describes the tables in the database owned by the current user in the Hive metastore.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2 (4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive database
TABLE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive table
LOCATION	VARCHAR2 (4000)		
NO_OF_COLS	NUMBER		Number of columns in the Hive table
CREATION_TIME	DATE		Time when the table was created
LAST_ACCESSED_TIME	DATE		Time of most recent access
OWNER	VARCHAR2 (4000)		Owner of the Hive table
TABLE_TYPE	VARCHAR2 (4000)	NOT NULL	Type of Hive table, such as external or managed
PARTITIONED	VARCHAR2 (4000)		Whether the table is partitioned (YES) or not (NO)
NO_OF_PART_KEYS	NUMBER		Number of partitions
INPUT_FORMAT	VARCHAR2 (4000)		Input format
OUTPUT_FORMAT	VARCHAR2 (4000)		Output format
SERIALIZATION	VARCHAR2 (4000)		SerDe serialization information
COMPRESSED	NUMBER		Whether the table is compressed (YES) or not (NO)

Column	Datatype	NULL	Description
HIVE_URI	VARCHAR2 (4000)		Hive database URI

See Also:

- [“DBA_HIVE_TABLES”](#)
 - [“USER_HIVE_TABLES”](#)
-
-

5.1.3.3 ALL_HIVE_COLUMNS

ALL_HIVE_COLUMNS describes the columns of all Hive tables accessible to the current user.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See [“About the Common Directory”](#).

Related Views

- DBA_HIVE_COLUMNS describes the columns of all tables in the Hive metastore.
- USER_HIVE_COLUMNS describes the columns of the tables in the Hive database owned by the current user.

Column	Datatype	NULL	Description
CLUSTER_ID	VARCHAR2 (4000)	NOT NULL	Hadoop cluster where the Hive metastore is located
DATABASE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive database; if blank, then the default database
TABLE_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive table
COLUMN_NAME	VARCHAR2 (4000)	NOT NULL	Name of the Hive column
HIVE_COLUMN_TYPE	VARCHAR2 (4000)	NOT NULL	Data type of the Hive column
ORACLE_COLUMN_TYPE	VARCHAR2 (4000)	NOT NULL	Oracle data type equivalent to Hive data type
LOCATION	VARCHAR2 (4000)		
OWNER	VARCHAR2 (4000)		Owner of the Hive table
CREATION_TIME	DATE		Time when the table was created
HIVE_URI	VARCHAR2 (4000)		Hive database URI

See Also:

- [“DBA_HIVE_COLUMNS”](#)
 - [“USER_HIVE_COLUMNS”](#)
-

5.1.3.4 DBA_HIVE_DATABASES

DBA_HIVE_DATABASES describes all the databases in the Hive metastore. Its columns are the same as those in ALL_HIVE_DATABASES.

See Also:

[“ALL_HIVE_DATABASES”](#)

5.1.3.5 DBA_HIVE_TABLES

DBA_HIVE_TABLES describes all tables in the Hive metastore. Its columns are the same as those in ALL_HIVE_TABLES.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. See [“About the Common Directory”](#).

See Also:

[“ALL_HIVE_TABLES”](#)

5.1.3.6 DBA_HIVE_COLUMNS

DBA_HIVE_COLUMNS describes the columns of all tables in the Hive metastore. Its columns are the same as those in ALL_HIVE_COLUMNS.

See Also:

[“ALL_HIVE_COLUMNS”](#)

5.1.3.7 USER_HIVE_DATABASES

USER_HIVE_DATABASES describes the databases in the Hive metastore owned by the current user. Its columns (except for OWNER) are the same as those in ALL_HIVE_DATABASES.

See Also:

[“ALL_HIVE_DATABASES”](#)

5.1.3.8 USER_HIVE_TABLES

USER_HIVE_TABLES describes the tables in the database owned by the current user in the Hive metastore. Its columns (except for OWNER) are the same as those in ALL_HIVE_TABLES.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See “[About the Common Directory](#)”.

See Also:

“[ALL_HIVE_TABLES](#)”

5.1.3.9 USER_HIVE_COLUMNS

USER_HIVE_COLUMNS describes the columns of the tables in the Hive database owned by the current user. Its columns (except for OWNER) are the same as those in ALL_HIVE_COLUMNS.

The Oracle Big Data SQL configuration must identify the default Hive database for the current user. The current user must also have READ privileges on the ORA_BIGSQL_CONFIG database directory. See “[About the Common Directory](#)”.

See Also:

“[ALL_HIVE_COLUMNS](#)”

Appendices

Licensing Information

A.1 Oracle Big Data SQL

The licensing for Oracle Big Data SQL is separate from the licensing for other Oracle products.

When you purchase a license for Oracle Big Data SQL, note the following:

- A separate license must be procured per disk per Hadoop cluster.
- All nodes within the Hadoop cluster that runs Oracle Big Data SQL must be licensed.
- All disks within every node that is part of a cluster running Oracle Big Data SQL must be licensed. Partial licensing within a node is not available. All nodes in the cluster are included.
- Only the Hadoop cluster side (Oracle Big Data Appliance, or other) of an Oracle Big Data SQL installation is licensed and no additional license is required for the database server side.
- Oracle Copy to Hadoop licensing is included.
- Oracle Super Cluster is not included.

Index

A

access drivers, [1-1](#), [4-3](#)
ACCESS PARAMETERS clause
 special characters, [5-3](#)
 syntax rules, [5-3](#)
ACCESS PARAMETERS Clause
 syntax, [5-3](#)
ALL_HIVE_COLUMNS view, [5-18](#)
ALL_HIVE_DATABASES view, [5-16](#)
ALL_HIVE_TABLES view, [3-2](#), [5-17](#)
array overflows, [5-13](#)
Audit Vault
 plug-in configuration, [2-8](#)

B

binary overflows, [5-13](#)

C

catalog views, [5-16](#)
character overflows, [5-13](#)
column mapping, [5-6](#)
com.oracle.bigdata.colmap, [5-6](#)
com.oracle.bigdata.datamode, [5-7](#)
com.oracle.bigdata.erroropt, [5-8](#)
com.oracle.bigdata.fields, [5-9](#)
com.oracle.bigdata.fileformat, [5-10](#)
com.oracle.bigdata.log.exec, [5-11](#)
com.oracle.bigdata.log.qc, [5-12](#)
com.oracle.bigdata.overflow, [5-13](#)
com.oracle.bigdata.rowformat, [5-14](#)
com.oracle.bigdata.tablename, [5-15](#)
common directory, [3-17](#)
CREATE TABLE ORGANIZATION EXTERNAL
 syntax, [3-13](#), [4-4](#)
CREATE TABLE statement
 generating automatically for Hive, [5-1](#)
CREATE_EXTDDL_FOR_HIVE function
 syntax, [5-1](#)

D

data dictionary views, [5-16](#)
data mode, [5-7](#)
data source name, [5-15](#)
data type conversion (Big Data SQL), [3-15](#)
data types (HDFS), [5-9](#)
DBA_HIVE_COLUMNS view, [5-19](#)
DBA_HIVE_DATABASES view, [5-19](#)
DBA_HIVE_TABLES view, [5-19](#)
DBMS_HADOOP package, [5-1](#)
DBMS_OUTPUT package, [3-2](#)
DEFAULT DIRECTORY clause, [3-14](#)
delimited text files, [5-14](#)

E

error handling, [5-8](#)
error handling (Big Data SQL), [3-17](#)
external tables
 about, [1-1](#), [4-3](#)

F

field extraction, [5-14](#)
field names, [5-9](#)

H

Hadoop log files, [5-11](#)
HDFS Transparent Encryption, [2-8](#)
Hive columns, [5-18](#)
Hive data
 access from Oracle Database, [3-1](#)
Hive databases, [5-16](#)
Hive table sources, [5-15](#)
Hive tables, [5-17](#)
Hive views, [5-16](#)

L

LOCATION clause, [3-14](#)
log files, [5-12](#)

O

- Oracle Audit Vault and Database Firewall
 - plug-in configuration, [2-8](#)
- Oracle Big Data SQL
 - access drivers, [1-1](#)
 - data type conversion, [3-15](#)
 - general description, [1-1](#)
 - installation changes on the Oracle Database server ., [3-17](#)
 - security, [2-8](#)
- Oracle Database
 - access to Hive data, [3-1](#)
- Oracle Exadata Machine
 - Big Data SQL installation changes, [3-17](#)
- ORACLE_HDFS access driver, [3-11](#)
- ORACLE_HIVE
 - access parameters, [5-5](#)
- ORACLE_HIVE examples, [3-3](#)
- ORC files, [5-10](#)
- overflow handling, [5-13](#)

P

- Parquet files, [5-10](#)
- parsing HDFS files, [5-14](#)
- PL/SQL packages, [5-1](#)
- PUT_LINE function, [3-2](#)

R

- RC files, [5-10](#)

- REJECT LIMIT clause, [3-15](#)
- row format description, [5-10](#)
- row formats, [5-14](#)

S

- sequence files, [5-10](#)
- SerDe parsing, [5-14](#)
- SmartScan, [1-3](#)
- SmartScan mode, [5-7](#)
- source name, [5-15](#)
- static data dictionary views, [5-16](#)
- struct overflows, [5-13](#)

T

- text files, [5-10](#)
- text overflows, [5-13](#)
- TYPE clause, [3-14](#)

U

- union overflows, [5-13](#)
- user access from Oracle Database, [3-16](#)
- USER_HIVE_COLUMNS view, [5-20](#)
- USER_HIVE_DATABASES view, [5-19](#)
- USER_HIVE_TABLES view, [5-20](#)