

Course Guide

# **Big Data Engineer 2021**

## ***IBM Db2 Big SQL***

Course code SABSQ ERC 3.0

Ahmed Abdel-Baky

Maria Farid

Heba Aboulmagd

Abdelrahman Hassan

Mohamed El-Khouly

Norhan Khaled

Adel El-Metwally

Ramy Said

Nouran El-Sheikh

Dina Sayed



## February 2021 edition

### Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

### Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

**© Copyright International Business Machines Corporation 2016, 2021.**

**This document may not be reproduced in whole or in part without the prior written permission of IBM.**

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Trademarks .....</b>	<b>viii</b>
<b>Course description .....</b>	<b>ix</b>
<b>Agenda .....</b>	<b>xi</b>
<b>Unit 1. Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data .....</b>	<b>1-1</b>
Unit objectives .....	1-2
1.1. Introduction to Db2 Big SQL .....	1-3
Introduction to Db2 Big SQL .....	1-4
Topics .....	1-5
Db2 Big SQL is SQL on Hadoop .....	1-6
Why use SQL access for Hadoop? .....	1-8
What does Db2 Big SQL provide? .....	1-9
Db2 Big SQL provides comprehensive and standard SQL .....	1-10
Db2 Big SQL provides powerful optimization and performance .....	1-11
Db2 Big SQL supports various storage formats .....	1-12
Db2 Big SQL integrates with RDBMSs .....	1-14
Db2 Big SQL architecture .....	1-16
The relationship between Db2 Big SQL and Db2 .....	1-19
1.2. Accessing and managing Db2 Big SQL .....	1-22
Accessing and managing Db2 Big SQL .....	1-23
Topics .....	1-24
Starting and stopping Db2 Big SQL by using Ambari .....	1-25
Starting and stopping Db2 Big SQL by using a UDF .....	1-26
Accessing Db2 Big SQL .....	1-27
JSqlsh (1 of 3) .....	1-29
JSqlsh (2 of 3) .....	1-30
JSqlsh (3 of 3) .....	1-31
Web tools through the Db2 Big SQL console .....	1-32
Unit summary .....	1-33
Review questions .....	1-34
Review questions (cont.) .....	1-35
Review answers .....	1-36
Review answers (cont.) .....	1-37
Exercise: Connecting to the IBM Db2 Big SQL server .....	1-38
Exercise objectives .....	1-39
<b>Unit 2. Creating IBM Db2 Big SQL schemas and tables .....</b>	<b>2-1</b>
Unit objectives .....	2-2
2.1. Db2 Big SQL tables and schemas .....	2-3
Db2 Big SQL tables and schemas .....	2-4
Topics .....	2-5
Db2 Big SQL terminologies .....	2-6
Partitioned tables .....	2-8
Partitions: A picture says 1,000 words .....	2-9
Creating Db2 Big SQL schemas .....	2-10
Using the web GUI to browse the HDFS .....	2-12
Creating a Db2 Big SQL table .....	2-13
Results from previous CREATE TABLE .....	2-15

More about CREATE TABLE .....	2-17
CREATE TABLE: Partitioned tables .....	2-18
More CREATE TABLE features .....	2-20
CREATE VIEW .....	2-22
Loading data into Db2 Big SQL tables .....	2-23
Populating Db2 Big SQL tables by using LOAD .....	2-25
Populating Db2 Big SQL tables by using INSERT (1 of 2) .....	2-26
Populating Db2 Big SQL tables by using INSERT (2 of 2) .....	2-27
Populating Db2 Big SQL tables by using CREATE TABLE ... AS SELECT .....	2-28
2.2. Data types that are supported by Db2 Big SQL .....	2-29
Data types that are supported by Db2 Big SQL .....	2-30
Topics .....	2-31
Data types .....	2-32
Data types (cont.) .....	2-34
Data type mapping .....	2-36
REAL and FLOAT types .....	2-37
STRING type .....	2-38
Unit summary .....	2-40
Review questions .....	2-41
Review questions (cont.) .....	2-42
Review answers .....	2-43
Review answers (cont.) .....	2-44
Exercise: Creating and managing Db2 Big SQL schemas and tables .....	2-45
Exercise objectives .....	2-46
<b>Unit 3. File formats and querying IBM Db2 Big SQL tables .....</b>	<b>3-1</b>
Unit objectives .....	3-2
3.1. File formats and Db2 Big SQL .....	3-3
File formats and Db2 Big SQL .....	3-4
Topics .....	3-5
File formats and Db2 Big SQL .....	3-6
Text (1 of 5) .....	3-8
Text (2 of 5) .....	3-10
Text (3 of 5) .....	3-12
Text (4 of 5) .....	3-13
Text (5 of 5) .....	3-15
Sequence (1 of 2) .....	3-17
Sequence (2 of 2) .....	3-19
Parquet (1 of 4) .....	3-20
Creating a table with underlying Parquet files (2 of 4) .....	3-22
Loading a Parquet table (3 of 4) .....	3-24
Parquet and compression (4 of 4) .....	3-26
Optimized Row Columnar (1 of 2) .....	3-28
ORC and compression (2 of 2) .....	3-30
Record Columnar .....	3-31
Avro (1 of 5) .....	3-33
Creating an Avro table with an inline Avro schema (2 of 5) .....	3-35
Creating an Avro table: external schema (3 of 5) .....	3-38
Avro schema data type mapping (4 of 5) .....	3-40
Avro schema example (5 of 5) .....	3-42
3.2. Querying Db2 Big SQL tables .....	3-43
Querying Db2 Big SQL tables .....	3-44
Topics .....	3-45
Power of standard SQL .....	3-46
SQL capability highlights (1 of 4) .....	3-47
SQL capability highlights (2 of 4) .....	3-49

SQL capability highlights example (3 of 4) . . . . .	3-51
SQL capability highlights (4 of 4) . . . . .	3-52
A word about SQL compatibility . . . . .	3-54
Functions . . . . .	3-56
Unit summary . . . . .	3-58
Review questions . . . . .	3-59
Review questions (cont.) . . . . .	3-60
Review answers . . . . .	3-61
Review answers (cont.) . . . . .	3-62
Exercise: Querying Db2 Big SQL tables . . . . .	3-63
Exercise objectives . . . . .	3-64
<b>Unit 4. Configuring IBM Db2 Big SQL security . . . . .</b>	<b>4-1</b>
Unit objectives . . . . .	4-2
4.1. Db2 Big SQL encryption features . . . . .	4-3
Db2 Big SQL encryption features . . . . .	4-4
Topics . . . . .	4-5
Basic process architecture . . . . .	4-6
Encryption features . . . . .	4-8
4.2. Db2 Big SQL authentication . . . . .	4-10
Db2 Big SQL authentication . . . . .	4-11
Topics . . . . .	4-12
Authentication for Db2 Big SQL . . . . .	4-13
Setting up Kerberos for the Db2 Big SQL service . . . . .	4-15
4.3. Apache Ranger security for Db2 Big SQL . . . . .	4-17
Apache Ranger security for Db2 Big SQL . . . . .	4-18
Topics . . . . .	4-19
Authorization that uses the Db2 Big SQL Apache Ranger plug-in . . . . .	4-20
Database operations that the Apache Ranger plug-in controls . . . . .	4-22
4.4. Native Db2 Big SQL authorization . . . . .	4-23
Native Db2 Big SQL authorization . . . . .	4-24
Topics . . . . .	4-25
Authorization of Db2 Big SQL objects . . . . .	4-26
Level 1: Controlling access with authorization in the distributed file system . . . . .	4-27
Level 2: Authorization with the GRANT command . . . . .	4-29
Level 3: Authorization at the row and column level . . . . .	4-30
Level 3: Why use row and column access control . . . . .	4-31
Level 3: Advantages of row and column access control . . . . .	4-32
Level 3: Row-based access control (1 of 5) . . . . .	4-34
Level 3: Row-based access control (2 of 5) . . . . .	4-35
Level 3: Row-based access control (3 of 5) . . . . .	4-37
Level 3: Row-based access control (4 of 5) . . . . .	4-38
Level 3: Row-based access control (5 of 5) . . . . .	4-39
Level 3: Column-based access control (1 of 5) . . . . .	4-40
Level 3: Column-based access control (2 of 5) . . . . .	4-41
Level 3: Column-based access control (3 of 5) . . . . .	4-42
Level 3: Column-based access control (4 of 5) . . . . .	4-43
Level 3: Column-based access control (5 of 5) . . . . .	4-44
Level 4: Controlling access by using VIEWS or STORED PROCEDURES . . . . .	4-45
4.5. Impersonation in Db2 Big SQL . . . . .	4-46
Impersonation in Db2 Big SQL . . . . .	4-47
Topics . . . . .	4-48
Impersonation in Db2 Big SQL . . . . .	4-49
Why would you want to use impersonation . . . . .	4-51
Basic architecture with impersonation . . . . .	4-52
Populating Apache Hive and Db2 Big SQL tables by using HDFS directly . . . . .	4-53

Enabling impersonation for Db2 Big SQL .....	4-54
Enabling impersonation in Apache Hive (1 of 2) .....	4-56
Enabling impersonation in Apache Hive (2 of 2) .....	4-58
What is UMASK .....	4-59
Examples .....	4-60
Impersonation usage notes and restrictions .....	4-61
Unit summary .....	4-63
Review questions .....	4-64
Review questions (cont.) .....	4-65
Review answers .....	4-66
Review answers (cont.) .....	4-67
Exercise: Configuring authorizations of Db2 Big SQL objects .....	4-68
Exercise objectives .....	4-69
Exercise: Configuring impersonation in Db2 Big SQL .....	4-70
Exercise objectives .....	4-71
<b>Unit 5. Data federation with IBM Db2 Big SQL.....</b>	<b>5-1</b>
Unit objectives .....	5-2
5.1. Db2 Big SQL federation overview .....	5-3
Db2 Big SQL federation overview .....	5-4
Topics .....	5-5
"Build it, and they will come." .....	5-6
Is there a better way? .....	5-8
Solution: Db2 Big SQL with Fluid Query .....	5-9
Db2 Big SQL federation overview .....	5-10
Federated system .....	5-12
Supported data sources .....	5-14
Example: SELECT .....	5-16
Federated system main components .....	5-17
5.2. Db2 Big SQL federation components .....	5-19
Db2 Big SQL federation components .....	5-20
Topics .....	5-21
Federation components (1 of 5) .....	5-22
Federation components (2 of 5): Wrappers .....	5-23
Federation components (3 of 5): Server definitions .....	5-25
Federation components (3 of 5): Server definitions (cont'd) .....	5-27
Federation components (4 of 5): User mappings .....	5-28
Federation components (4 of 5): User mappings (cont'd) .....	5-30
Federation components (5 of 5): Nicknames .....	5-31
Federated three-part names .....	5-33
Unit summary .....	5-34
Review questions .....	5-35
Review questions (cont.) .....	5-36
Review answers .....	5-37
Review answers (cont.) .....	5-38
Exercise: Getting started with Db2 Big SQL federation .....	5-39
Exercise objectives .....	5-40

---

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

Db2®  
Power®

IBM Cloud™  
PureData®

IBM PureData®

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Netezza® and NPS® are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Other product and service names might be trademarks of IBM or other companies.

---

# Course description

## **Big Data Engineer 2021 - *IBM Db2 Big SQL***

**Duration: 2 days**

### **Purpose**

In this course, you learn about Db2 Big SQL, and how to use it to access data that resides in the Hadoop Distributed File System (HDFS). You create Db2 Big SQL schemas and tables and query tables. You explore Db2 Big SQL security and learn how to configure authorizations of Db2 Big SQL objects, and impersonation in Db2 Big SQL. Finally, this course introduces federation with Db2 Big SQL and you run queries from a federated Db2 Big SQL server to query the data on a Db2 service instance on IBM Cloud.

### **Audience**

Undergraduate senior students from IT related academic programs, for example, computer science, software engineering, information systems and others.

### **Prerequisites**

Successful completion of course “Big Data Engineer 2021 - Big Data Ecosystem”.

## Objectives

After completing this course, you should be able to:

- Describe Db2 Big SQL.
- Explain how Db2 Big SQL fits in the Hadoop architecture.
- Start and stop Db2 Big SQL by using Ambari and the command line interface (CLI).
- Connect to a Db2 Big SQL server and run Db2 Big SQL queries.
- Use the Db2 Big SQL console.
- Create Db2 Big SQL schemas and tables.
- Load data into Db2 Big SQL tables following best practices.
- Describe file formats that are supported by Db2 Big SQL
- Query Db2 Big SQL tables by using various Data Manipulation Languages (DMLs)
- Work with ARRAY data type.
- Create user-defined functions (UDFs).
- Configure Db2 Big SQL security.
- Configure impersonation in Db2 Big SQL.
- Explain federation with Db2 Big SQL.
- Set up and configure a federated server to use different data sources.
- Run queries from the federated Db2 Big SQL server to query the data on a Db2 service instance on IBM Cloud.

---

# Agenda

---

**Note**

The following unit and exercise durations are estimates, and might not reflect every class experience.

---

## Day 1

- (00:30) Welcome
- (01:00) Unit 1 - Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data
- (01:00) Exercise 1 - Connecting to the IBM Db2 Big SQL server
- (01:00) Lunch break
- (01:00) Unit 2 - Creating IBM Db2 Big SQL schemas and tables
- (01:30) Exercise 2 - Creating and managing Db2 Big SQL schemas and tables

## Day 2

- (01:00) Unit 3 - File formats and querying IBM Db2 Big SQL tables
- (01:00) Exercise 3 - Querying Db2 Big SQL tables
- (01:00) Unit 4 - Configuring IBM Db2 Big SQL security
- (00:20) Exercise 4 - Configuring authorizations of Db2 Big SQL objects (Demo only)
- (00:20) Exercise 5 - Configuring impersonation in Db2 Big SQL (Demo only)
- (01:00) Lunch break
- (00:45) Unit 5 - Data federation with IBM Db2 Big SQL
- (01:00) Exercise 6 - Getting started with Db2 Big SQL federation

# Unit 1. Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

## Estimated time

01:00

## Overview

In this unit, you learn about Db2 Big SQL, and how to use it to access data that resides in the Hadoop Distributed File System (HDFS).

## Unit objectives

- Describe Db2 Big SQL.
- Explain how Db2 Big SQL fits into the Hadoop architecture.
- Start and stop Db2 Big SQL by using Ambari and the command-line interface (CLI).
- Connect to Db2 Big SQL by using the CLI.
- Connect to Db2 Big SQL by using the Db2 Big SQL console.

## 1.1. Introduction to Db2 Big SQL

## Introduction to Db2 Big SQL

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-2. Introduction to Db2 Big SQL

## Topics

- ▶ Introduction to Db2 Big SQL
  - Accessing and managing Db2 Big SQL

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-3. Topics*

## Db2 Big SQL is SQL on Hadoop

- Db2 Big SQL integrates with the Apache Hive foundation:
  - Integrates with the Hive metastore.
  - Instead of MapReduce, uses a powerful native C/C++ MPP engine.
- View your data in the Hadoop file system.
- No proprietary storage format.
- Compliant with industry standards for SQL.
- The same SQL can be used on your warehouse data with little or no modifications.

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-4. *Db2 Big SQL is SQL on Hadoop*

Db2 Big SQL uses HCatalog (and the Hive Metastore) as its underlying data representation and access method.

Apache Hive is an open source data warehouse software for reading, writing, and managing large data set files that are stored directly in either the Apache Hadoop Distributed File System (HDFS) or other data storage systems, such as Apache HBase. Hive consists of an execution engine, a storage model, and a metastore. This metastore tracks all the metadata about tables within Hadoop and is used by multiple projects within the Hadoop infrastructure.

Db2 Big SQL provides an alternative execution engine, which has many advantages compared to Hive to accommodate various workloads. Db2 Big SQL shines with high concurrency workloads, complex SQL, data virtualization, and more. Rather than using MapReduce (which Hive uses), Db2 Big SQL uses a powerful, native massively parallel processing (MPP) engine, which is written in C/C++. Db2 Big SQL shares metadata with Hive by using the same data on disk and same definition of tables within the metastore. A table in Hive can be seen in Db2 Big SQL and vice-versa.

Think of Db2 Big SQL as a logical view on top of your Hadoop data. The data is in Hadoop, and all you must do is use Db2 Big SQL to access it. There is no need to change the format or migrate the data out of Hadoop to do any work on the data. Db2 Big SQL is compliant with industry standards for SQL, cryptography, and GDPR. So, when you migrate your data into Hadoop, the same SQL can be used on your warehouse data with little or no modification. That is one of the significant benefits of Db2 Big SQL: There is no need to learn anything new, and you can use your existing queries.

There are several tools that are available to work with data on Hadoop, such as MapReduce, Pig, and Hive. MapReduce, being one of the most common tools, is highly scalable but difficult to use. There is much coding that is required for running batch jobs. There are other tools, but they require a specific set of expertise. What Db2 Big SQL offers is a common and familiar query syntax that everybody understands.

## References:

- Db2 Big SQL integration with Hive:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...l/doc/bsql\\_data\\_types.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...l/doc/bsql_data_types.html)
- Hive introduction:  
<https://www.ibm.com/analytics/hadoop/hive>
- Db2 Big SQL introduction:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...l/doc/overview\\_icnav.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...l/doc/overview_icnav.html)  
<https://www.ibm.com/products/db2-big-sql>
- Standards compliance:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...l/doc/stds\\_compl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...l/doc/stds_compl.html)

## Why use SQL access for Hadoop?

- Data warehouse modernization is a leading Hadoop use case:
  - Offload "cold" warehouse data onto a query-ready Hadoop platform.
  - Explore, transform, analyze, and aggregate social media data, log records, and other data, and upload summary data to a warehouse.
- Limited availability of skills in MapReduce, Pig, and other programs.
- SQL opens the data to a much wider audience:
  - It uses a familiar, widely known syntax.
  - It has a common catalog for identifying data and structure.

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-5. Why use SQL access for Hadoop?

There is a good motivation to build a new SQL engine for Hadoop.

Hadoop appeals to businesses that must store large volumes of various types of data in a cost-effective way. It is easy to store the data in a Hadoop cluster and scale out inexpensively, but it is difficult to get value out of that information. The expertise that is required to do traditional Hadoop programming is in short supply and expensive. But, the potential for Hadoop to extend the warehouse is so compelling that SQL access to Hadoop data is prevalent. You can have a flexible platform like Hadoop and an entire industry with SQL skills and extensive tools that are based on SQL. If Hadoop clusters have that native SQL interface, it rapidly speeds up Hadoop's adoption, and make many analytics opportunities possible.

### Reference:

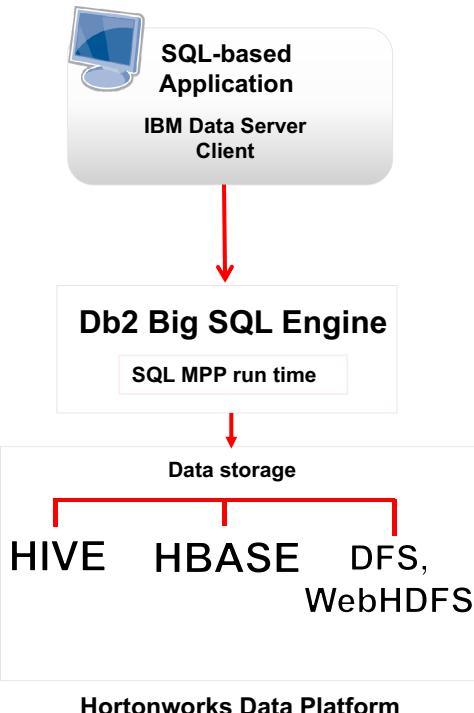
Need for SQL access for Hadoop:

<https://www.ibm.com/developerworks/library/ba-compare-sql-access-hadoop/index.html>



## What does Db2 Big SQL provide?

- Comprehensive, standard SQL
- Optimization and performance
- Support for various storage formats
- Integration with RDBMSs



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-6. What does Db2 Big SQL provide?

Db2 Big SQL is a hybrid, high-performance MPP SQL engine for Hadoop with support for various data sources, including HDFS, RDBMS, NoSQL databases, object stores, and WebHDFS. Db2 Big SQL uses a single database connection or single query for best-in-class analytic capabilities.

Db2 Big SQL offers low latency queries, security, SQL compatibility, and federation capabilities, which enable organizations to derive value from enterprise data.

### References:

- Db2 Big SQL overview:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_icnav.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_icnav.html)
- Db2 Big SQL Performance capabilities:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_performance.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_performance.html)
- Db2 Big SQL query can access various data sources:  
<https://www.ibmbigdatahub.com/blog/real-scoop-enterprise-hadoop>

## Db2 Big SQL provides comprehensive and standard SQL

- **SELECT:** Joins, unions, aggregates, subqueries, and other functions
- **UPDATE/DELETE** (HBase-managed tables)
- **GRANT/REVOKE, INSERT ... INTO**
- SQL procedural logic (SQL PL)
- Stored procedures and user-defined functions (UDFs)
- IBM data server JDBC and ODBC drivers

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-7. Db2 Big SQL provides comprehensive and standard SQL*

Db2 Big SQL provides SQL developers with a way for querying data that is managed by Hadoop. Most of the SQL that you use with your RDBMs can be used with Db2 Big SQL. In addition, a *LOAD* command enables administrators to populate Db2 Big SQL tables with data from various sources. Also, Db2 Big SQL JDBC and ODBC drivers enable many existing tools to use Db2 Big SQL to query this distributed data.

### References:

- Db2 Big SQL MPP SQL engine provides you with several SQL processing features:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/overview\\_processing.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/overview_processing.html)
- Db2 Big SQL allows working with different data sources:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/bigsq...\\_analyzingbigdata.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/bigsq..._analyzingbigdata.html)
- Db2 Big SQL supports for SQL Procedural Language (SQL PL):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bsql\\_reference.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bsql_reference.html)

## Db2 Big SQL provides powerful optimization and performance

- IBM MPP engine (native C++) replaces the Java MapReduce layer.
- Continuous running daemons (no startup latency).
- Message passing allows data to flow between nodes without persisting intermediate results.
- In-memory operations that can spill to disk (useful for aggregations, and sorts that exceed the available RAM).
- Cost-based query optimization.

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-8. Db2 Big SQL provides powerful optimization and performance*

Db2 Big SQL is designed for performance. The Db2 Big SQL runtime execution engine is native code (C/C++). It replaces MapReduce by using a modern MPP architecture. The compiler and run time are written in native code (C/C++), so it is much faster. The SQL engine pushes down the processing to the same node the holds the data, so all the processing happens locally at the data. There is also no startup latency because the daemons are continuously running. The operations occur in memory, and if necessary, the memory can spill over to disk for processing so that aggregations and sorts can exceed the available RAM.

### References:

- Performance capabilities:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_performance.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_performance.html)

- Data locality and compaction:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsqld\\_compaction.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsqld_compaction.html)

- The query optimizer:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint26.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint26.html)

## Db2 Big SQL supports various storage formats

- Text (delimited)
- Optimized Row Columnar (ORC)
- Parquet
- Avro
- Record Columnar (RC)
- Sequence

No IBM proprietary format is required.

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-9. Db2 Big SQL supports various storage formats

The Hadoop environment can read many storage formats. This flexibility is partially because of the INPUTFORMAT and OUTPUTFORMAT classes that you can specify on the CREATE and ALTER table statements and because of the use of installed and customized SerDes (Serializer/Deserializer) classes. The file formats that are listed here are available either by using explicit SQL syntax, such as STORED AS PARQUETFILE, or by using installed interfaces, such as Avro. Db2 Big SQL generally supports anything that Hadoop handles, including compression types, file formats, and SerDes, among others.

For common table formats, a native I/O engine is used. These table formats include SEQUENCEFILE, RCFILE, AVRO, PARQUET, and TEXTFILE. For all others, such as ORC, a Java I/O engine is used, which maximizes compatibility with existing tables and allows for custom file formats and SerDes.

For optimal performance and functions, the ORC file format is recommended for Db2 Big SQL.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
doc/biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Db2 Big SQL readers and writers:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
doc/bigsq\\_iolibraries.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Db2 Big SQL SerDes:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
com/msql/doc/doc/biga\\_serde\\_jar\\_sync.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Db2 Big SQL integrates with RDBMSs

- The Db2 Big SQL LOAD command can load data from a remote DB or table.
- You can query heterogeneous databases by using the federation feature.

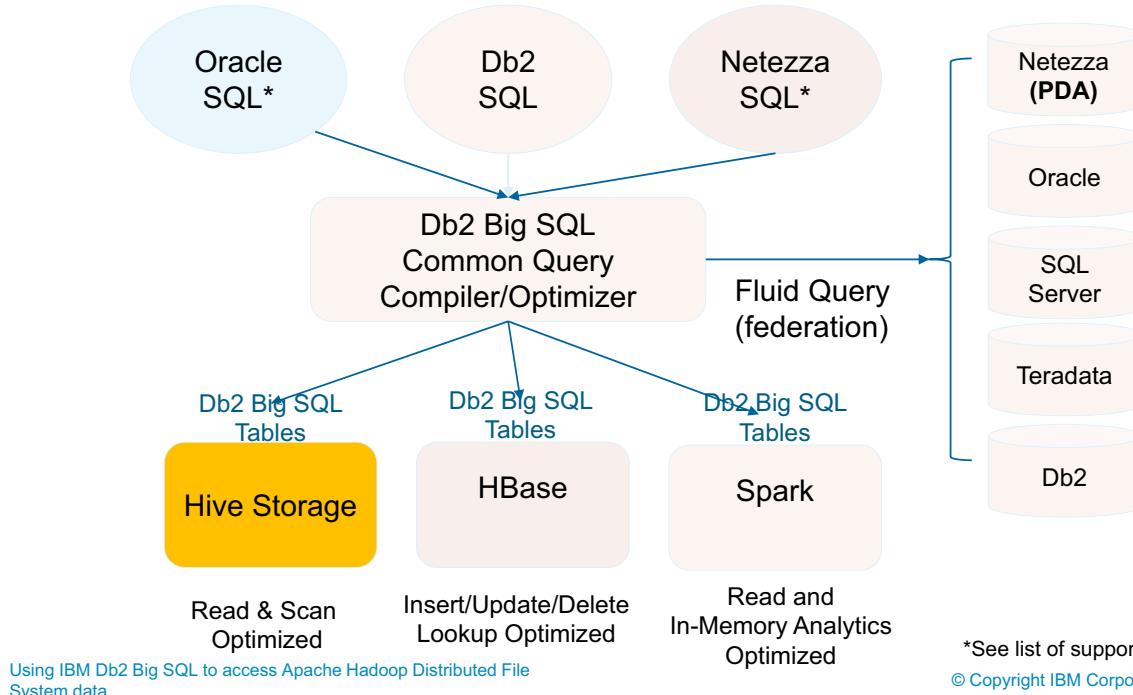


Figure 1-10. Db2 Big SQL integrates with RDBMSs

Db2 Big SQL can communicate with heterogeneous databases, such as Oracle or Teradata, by using the federation feature. Multi-vendor workloads are more convenient than ever.

Db2 Big SQL provides the following features:

- A unified view of all your tables, with federated query support to external databases.
- Stored optimally as Hive, HBase, or read with Spark (optimized for the expected workload).
- Secured under a single security model (including row/column security across all).
- The ability to join across all data sets by using standard ANSI SQL across all types of tables.
- Use Oracle, NZ, or Db2 extensions.
- Use a single database connection and driver.

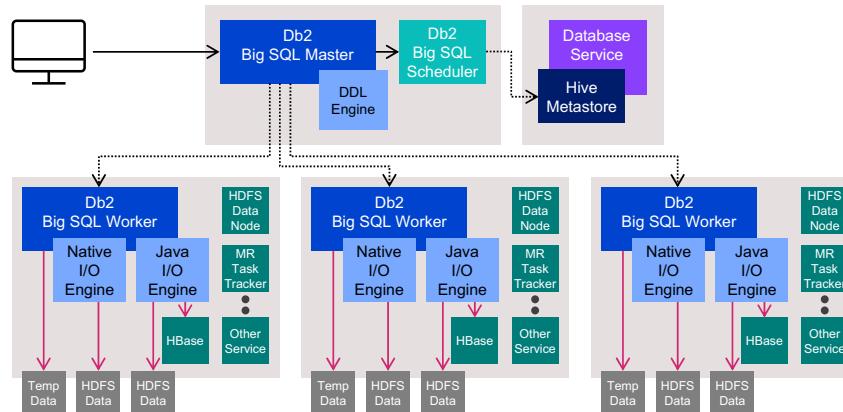
This integration makes Db2 Big SQL a unique SQL engine for Hadoop.

**References:**

- Federation capability:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
\[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\\_6.0.0/com.ibm.swg.im.bigsq...\]\(https://www.ibm.com/support/knowledgecenter/en/SSCRJT\_6.0.0/com.ibm.swg.im.bigsq...\)](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)
- Remote data sources that can be accessed by using the built-in federation capability:  
<https://www.ibm.com/support/pages/node/870650>
- Federated access to relational data:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Db2 Big SQL architecture

- Head (coordinator / management) node:
  - Listens to the JDBC/ODBC connections.
  - Compiles, optimizes, and coordinates the running of the query.
- Db2 Big SQL worker processes are on compute nodes (some or all).
- Worker nodes stream data between each other as needed.
- Workers can spill large data sets to local disk if needed.



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-11. Db2 Big SQL architecture

Db2 Big SQL is built on the IBM common SQL database technology. Db2 Big SQL is an MPP database engine that has all the standard RDBMS features and is optimized to work with the Apache Hadoop infrastructure.

The Db2 Big SQL server or service consists of one Db2 Big SQL head (two heads in an HA configuration) that is installed on a node that is called the *head node*, and multiple Db2 Big SQL workers that are installed on nodes that are called *worker nodes*.

### Db2 Big SQL head

The set of Db2 Big SQL processes that accept SQL query requests from applications and coordinate with Db2 Big SQL workers to process data and compute the results.

### Db2 Big SQL head node

The physical or virtual machine (VM) (node) on which the Db2 Big SQL head runs.

## **Db2 Big SQL worker**

The set of Db2 Big SQL processes that communicate with the Db2 Big SQL head to access data and compute query results. Db2 Big SQL workers are normally collocated with the HDFS DataNodes to facilitate local disk access. Db2 Big SQL can access and process HDFS data that conforms to most common Hadoop formats, such as Avro, Parquet, ORC, and Sequence. The Db2 Big SQL head coordinates the processing of SQL queries with the workers, which handle most of the HDFS data access and processing.

### **Db2 Big SQL worker node**

The physical or VM (node) on which the Db2 Big SQL worker runs.

### **Db2 Big SQL scheduler**

A process that runs on the Db2 Big SQL head node. The scheduler's function is to bridge the RDBMS domain and the Hadoop domain. The scheduler communicates with the Hive metastore to determine Hive table properties and schemas, and the HDFS NameNode to determine the location of file blocks. The scheduler responds to Db2 Big SQL head and worker requests for information about Hadoop data, including HDFS, HBase, object storage, and Hive metadata.

### **Db2 Big SQL metadata**

HDFS data properties such as name, location, format, and the relational schemas. This metadata, gathered through the scheduler, is used to enable consistent and optimal SQL processing of queries against HDFS data.

The following steps represent a simple overview of how Db2 Big SQL processes HDFS data:

1. Applications connect: Applications connect to the Db2 Big SQL head on the head node.
2. Queries are submitted: Queries that are submitted to the Db2 Big SQL head are compiled into optimized parallel execution plans by using the IBM common SQL engine's query optimizer.
3. Plans are distributed: The parallel execution plans are then distributed to Db2 Big SQL workers on the worker nodes.
4. Data is read and written: Workers have separate local processes called native HDFS readers and writers that read or write HDFS data in its stored format. A *Db2 Big SQL reader* is composed of Db2 Big SQL processes that run on the Db2 Big SQL worker nodes and read data at the request of Db2 Big SQL workers. Similarly, a *Db2 Big SQL writer* is composed of Db2 Big SQL processes that run on the Db2 Big SQL worker nodes and write data at the request of Db2 Big SQL workers. The workers communicate with these native readers or writers to access HDFS data when they process execution plans that they receive from the Db2 Big SQL head.
5. Predicates are applied: Native HDFS readers can apply predicates and project columns to minimize the amount of data that is returned to workers.

Db2 Big SQL uses the Hive Metastore (HCatalog) for table definitions, location, storage format, and encoding of input files. This Db2 Big SQL catalog is on the head node.

While the data is defined in the Hive Metastore and accessible in the Hadoop cluster, Db2 Big SQL can get to it. Db2 Big SQL stores some of the metadata from the Hive catalog locally for ease of access and to facilitate query execution.

Also, Db2 Big SQL workers can spill large data sets to local disk if needed, which allows Db2 Big SQL to work with data sets that are larger than the available memory.

**References:**

- Db2 Big SQL architecture:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/overview\\_icnav.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Db2 Big SQL uses HCatalog (and thus the Hive Metastore):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/bsql\\_data\\_types.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/bsql_data_types.html)

- Db2 Big SQL architecture image:

<https://medium.com/@ostechnote/ibm-big-sql-hdp-support-is-here-fb25d688e42b>

## The relationship between Db2 Big SQL and Db2

Db2 Big SQL and Db2 have the same "DNA":

- Bug fixes and enhancements (especially in Optimizer) in Db2 also benefit Db2 Big SQL.
- Enhancements that come from Db2 Big SQL are reintegrated into "Db2 Main" often.
- Features that are enabled for Db2 Big SQL for "almost free":
  - High availability and disaster recover (HADR) for the head node
  - Oracle PL/SQL support
  - Declared Global Temporary tables

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-12. The relationship between Db2 Big SQL and Db2*

Many Db2 technologies are in Db2 Big SQL:

- High availability and disaster recover (HADR) for the head node
- Oracle PL/SQL support
- Materialized Query Tables
- RUNSTATS command (Db2) is similar to the ANALYZE command (Db2 Big SQL)
- Row and column security
- Federation / Fluid Query
- Views
- SQL PL Stored Procedures UDFs
- Workload manager (WLM)
- System Temporary Table Spaces to support sort overflows
- User Temporary Table Spaces for Declared Global Temporary Tables

**References:**

- Db2 HADR:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_bigsq...ha.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_bigsq...ha.html)
  - Compatibility features for Oracle:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../c\\_compat\\_oracle.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../c_compat_oracle.html)
  - Hive transactional (atomicity, consistency, isolation, and durability (ACID)) tables:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bigsq...hive\\_acid\\_tables.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bigsq...hive_acid_tables.html)
  - Transactional behavior of Hadoop tables:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/bsql\\_trans\\_container.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/bsql_trans_container.html)
  - Materialized query tables:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/top...ics/iiyfed\\_tuning\\_mqt.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/top...ics/iiyfed_tuning_mqt.html)
  - CURRENT EXPLAIN SNAPSHOT:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/r0001109.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/r0001109.html)
  - Statistical views:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/c0059080.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/c0059080.html)
  - ANALYZE command:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/bsql\\_analyze.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/bsql_analyze.html)
  - Statistics that are used from expression columns in statistical views:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/c0059080.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/c0059080.html)
- Row and column access control (RCAC):
- [https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_rcac\\_overview.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_rcac_overview.html)
- Federation / Fluid Query:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../overview\\_federation.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../overview_federation.html)
  - CREATE VIEW:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/r0000935.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/r0000935.html)

- SQL Procedural Language (SQL PL):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0008419.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0008419.html)

- DB2GENERAL UDFs:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.db2.luw.apdv.routines.commsql.doc/doc/c0003369.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.db2.luw.apdv.routines.commsql.doc/doc/c0003369.html)

- Workload management:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_wlm\\_overview.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_wlm_overview.html)

- WLM:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview\\_performance.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview_performance.html)

- Temporary table space:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/admin\\_db\\_directories.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/admin_db_directories.html)

- Global Temporary table:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0003272.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0003272.html)

## 1.2. Accessing and managing Db2 Big SQL

## Accessing and managing Db2 Big SQL

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-13. Accessing and managing Db2 Big SQL

## Topics

- Introduction to Db2 Big SQL
- ▶ Accessing and managing Db2 Big SQL

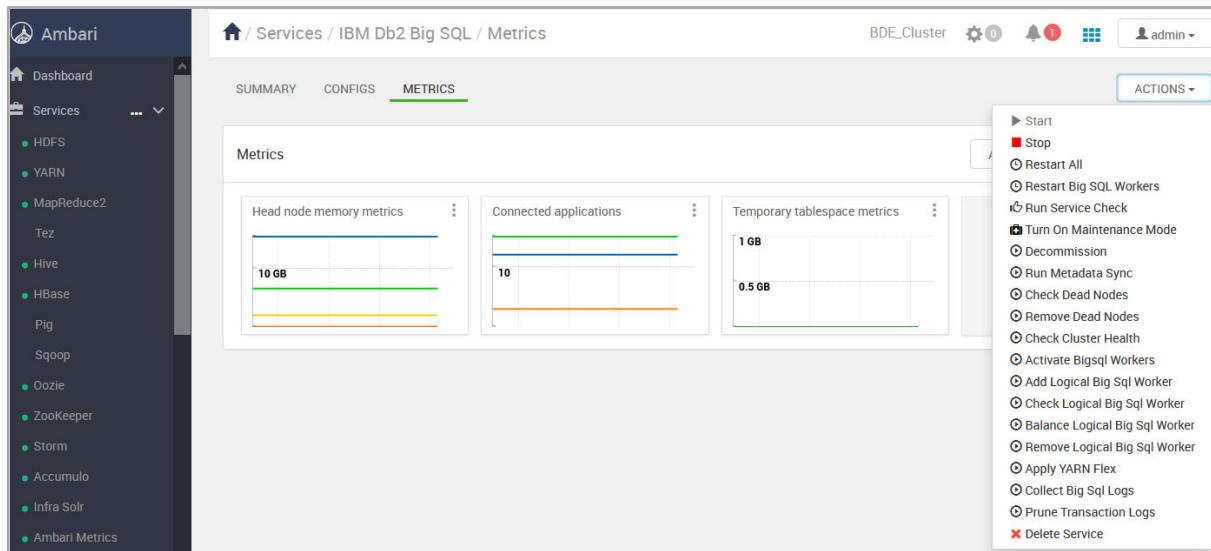
Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-14. Topics*

# IBM Training

## Starting and stopping Db2 Big SQL by using Ambari



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-15. Starting and stopping Db2 Big SQL by using Ambari

You can start or stop Db2 Big SQL by using either the Ambari interface or the CLI. To start Db2 Big SQL by using the Ambari web interface, go to **Db2 Big SQL** in the left pane, click **Service Actions**, and select either **Start** or **Stop**. You can also restart the Db2 Big SQL Workers, run a service check, or turn on maintenance mode from this menu.

### Reference:

Ambari dashboard:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_tools.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_tools.html)

## Starting and stopping Db2 Big SQL by using a UDF

As the `bigsq1` user, run the following commands from the active/primary head node:

- View the status of all Db2 Big SQL services:

```
$BIGSQL_HOME/bin/bigsq1 status
```

- Stop Db2 Big SQL:

```
$BIGSQL_HOME/bin/bigsq1 stop
```

- Start Db2 Big SQL:

```
$BIGSQL_HOME/bin/bigsq1 start
```

The screenshot shows the IBM Training interface with a blue header bar. Below it, a large blue section titled "Accessing Db2 Big SQL". To the right is the IBM logo.

**JSQSH SETUP WIZARD**

Welcome to the jsqsh setup wizard! This wizard provides a (crude) menu driven interface for managing several jsqsh configuration files. These files are all located in \$HOME/.jsqsh, and the name of the file being edited by a given screen will be indicated on the title of the screen

Note that many wizard screens require a relatively large console screen size, so you may want to resize your screen now.

(C)nection management wizard  
The connection management wizard allows you to define named connections using any JDBC driver that jsqsh recognizes. Once defined, jsqsh only

```
[dataengineer.ibm.com] [bigsql] 1> select tabschema, tablename from syscat.tables
[dataengineer.ibm.com] [bigsql] 2> fetch first 5 rows only;
+-----+-----+
| TABSCHEMA | TABNAME
+-----+-----+
| BIGSQL   | SMOKE_HADOOP2_1525422353
| BIGSQL   | SMOKE_HADOOP2_1639502892
| IBMCONSOLE | ALERT
| IBMCONSOLE | ALERT_PROPERTIES
| IBMCONSOLE | BLOCKING_LOG_ALERTS
+-----+
5 rows in results(first row: 0.014s; total: 0.016s)
[dataengineer.ibm.com] [bigsql] 1>
```

**IBM Db2 Big SQL**

RUN SQL

\* SYSCAT-Tables.sql \* SYSCAT-Columns... x

```
1 select tabschema, colname, colno, typename, length
2 from syscat.columns
3 fetch first 50 rows only
```

Syntax assistant

All results

Script	Date	Status	Runtime
SYSCAT-Tables.sql	09/09/20 23:25:33	✓ 1	0.015s
select tabschema, tablename from syscat.tables		✓ 1	0.015s
SYSCAT-Columns...09/09/20 23:25:14	09/09/20 23:25:14	✓ 1	0.027s
select tabschema, colname, colno, typename, length from sys...		✓ 1	0.027s
SYSTAB.sql	09/09/20 23:23:35	✓ 1	0.008s
Untitled - 1	09/09/20 23:20:44	✓ 1	0.046s

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-17. Accessing Db2 Big SQL

Db2 Big SQL includes a CLI that is called Java Sqshell (JSqsh). JSqsh is an open source database query tool that has many of the functions of a shell, such as variables, redirection, history, and CLI editing. It includes built-in help information and a wizard for establishing new database connections.

You can use the Db2 Big SQL console for database monitoring, administration, and configuration.

With the Db2 Big SQL console, administrators can manage schemas and tables, and developers can create and run Db2 Big SQL queries and UDFs in a simple and powerful user interface. The console also provides historical monitoring data and key health indicators, such as availability, responsiveness, and throughput.

Tools that support the IBM JDBC / ODBC driver are also options that can be used to access Db2 Big SQL.

**References:**

- Db2 Big SQL console:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_tools.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_tools.html)

- Highlights of the Db2 Big SQL console:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin\\_console\\_overview.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin_console_overview.html)

- Java SQL Shell (JSqsh):

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bsql\\_jsqsh.html#bsql\\_jsqsh](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bsql_jsqsh.html#bsql_jsqsh)

## JSqsh (1 of 3)

- Db2 Big SQL comes with a CLI that is called JSqsh that has the following features:
  - Open-source command client
  - Query history and query recall
  - Multiple result set display styles
  - Multiple active sessions
- You can start JSqsh by running the following command:  
`/usr/ibmpacks/common-utils/current/jsqsh/bin`

```
[bigsq@dataengineer /]$ cd /usr/ibmpacks/common-utils/current/jsqsh/bin
[bigsq@dataengineer bin]$ ls
jsqsh  jsqsh.bat
[bigsq@dataengineer bin]$ ./jsqsh
Welcome to JSqsh 6.0.0.3
Type "\help" for help topics. Using JLine.
WELCOME TO JSQSH!
```

Figure 1-18. JSqsh (1 of 3)

Db2 Big SQL has a CLI that is known as JSqsh, which is an open source client that works with any JDBC driver.

JSqsh can perform query history and query recall. It displays results in various styles depending on the file type, such as traditional, CSV, and JSON). You can also have multiple active sessions.

You can start the CLI by running the following command:

```
cd /usr/ibmpacks/common-utils/current/jsqsh/bin
./jsqsh
```

### References:

Accessing Db2 Big SQL from the JSqsh client:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsqql\\_jsqsh.html#bigsqql\\_jsqsh](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsqql_jsqsh.html#bigsqql_jsqsh)



## JSqsh (2 of 3)

- Run the JSqsh connection wizard to supply connection information:

```
JSQSH CONNECTION WIZARD - (edits $HOME/.jsqsh/connections.xml)
The following configuration properties are supported by this driver.

Connection name : bigsql
      Driver : IBM Data Server (DB2, Informix, Big SQL)
      JDBC URL : jdbc:db2://${server}:${port}/${db}

Connection URL Variables
-----
1           db : BIGSQL
2           port : 32051
3           server : dataengineer.ibm.com
4           user : bigsql
5           password :
6           Autoconnect : false

JDBC Driver Properties
-----
None

Enter a number to change a given configuration property, or
(T)est, (D)elete, (B)ack, (Q)uit, Add (P)roperty, or (S)ave:
```

```
[bigsql@dataengineer bin]$ ./jsqsh bigsql
Password:
Welcome to JSqsh 6.0.0.3
Type "\help" for help topics. Using JLine.
[dataengineer.ibm.com] [bigsql] 1>
```

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-19. JSqsh (2 of 3)

When you first use JSqsh, set up the connection by using the wizard to supply the information. Then, you can connect to the default bigsql database.

## JSqsh (3 of 3)

- The JSqsh default command terminator is a semicolon.
- The semicolon is also a valid SQL PL statement terminator.

```
CREATE FUNCTION COMM_AMOUNT(SALARY DEC(9,2))
RETURNS DEC(9,2)
LANGUAGE SQL
BEGIN ATOMIC
DECLARE REMAINDER DEC(9,2) DEFAULT 0.0;
...
END;
```

- JSqsh applies a basic heuristic to determine the actual statement end.

### Change the terminator.

```
1> \set terminator='@';
1> quit@
```

### Use the go command.

```
1> CREATE FUNCTION COMM_AMOUNT(SALARY
DEC(9,2))
...
20> END;
21> go
```

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-20. JSqsh (3 of 3)

If you used JSqsh before, you know that the default command terminator is a semicolon. With Db2 Big SQL, the semicolon is also a valid SQL PL statement terminator, so although JSqsh can take a best guess to determine the actual statement, it can sometimes get it wrong. When that happens, the statement does not run until you explicitly run the **go** command. The semicolon in Db2 Big SQL is the **go** command underneath. Alternatively, you can change the default terminator from a semicolon to another terminator, such as the @ symbol.

# IBM Training

## Web tools through the Db2 Big SQL console

The screenshot shows the Ambari web interface on the left and the IBM Db2 Big SQL console on the right. A callout box with an orange border and arrow points from the text "The web tool is started through the Ambari console." to the "Console UI" link in the Quick Links section of the Db2 Big SQL interface.

**Ambari Summary:**

- Components: Started (BIG SQL CONSOLE)

**Db2 Big SQL Console:**

- Quick Links:** Console UI (highlighted with an orange box and arrow)
- RUN SQL:**
  - Script: \* SYSCAT-Tables.sql
  - Code:

```
1 select tabschema, colname, colno, typename, length
2 from syscat.columns
3 fetch first 50 rows only
```
  - Run all
  - Remember my last behavior
- Result History:**

Script	Date	Status	Runtime
SYSCAT-Tables...	09/09/20 23:25:33	1	0.015s
SYSCAT-Column...	09/09/20 23:25:14	1	0.027s
SYSTAB.sql	09/09/20 23:23:35	1	0.008s
Untitled - 1	09/09/20 23:20:44	1	0.046s

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-21. Web tools through the Db2 Big SQL console

To extend the power of the Hortonworks Data Platform (HDP), install the Db2 Big SQL console, which is the tool for administering, monitoring, and managing Db2 Big SQL performance.

You can start the Db2 Big SQL console by either using the quick link in the Ambari web UI or by typing the following address into your web browser:

`https://<console_host>:<console_port>/console/`

Any valid Db2 Big SQL user can access the console after authentication. Console admin and user privileges can be granted or revoked by using Db2 Big SQL **GRANT** or **REVOKE** statements.

### References:

- Db2 Big SQL console installation:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/install\\_bigsq\\_console.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/install_bigsq_console.html)
- Highlights of Db2 Big SQL:  
[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin\\_console\\_overview.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin_console_overview.html)

## Unit summary

- Described Db2 Big SQL.
- Explained how Db2 Big SQL fits into the Hadoop architecture.
- Started and stopped Db2 Big SQL by using Ambari and the CLI.
- Connected to Db2 Big SQL by using the CLI.
- Connected to Db2 Big SQL by using the Db2 Big SQL console.

## Review questions

1. What are the reasons that your organization would use Db2 Big SQL?
  - A. Want to access your Hadoop data without using MapReduce.
  - B. It has superior SQL-on-Hadoop performance to optimize data ingestion and query performance for your enterprise.
  - C. No deep learning curve because it is compliant with industry standards for SQL.
  - D. No need to change the format or migrate the data out of Hadoop to do any work on the data.
  - E. All the above.
2. True or False: For optimal performance and function, the ORC file format is recommended for Db2 Big SQL.
3. True or False: You can access Db2 Big SQL through Db2 Big SQL console only.



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

*Figure 1-23. Review questions*

Write your answers here:

1. E. All the above.
2. True.
3. False, you can access Db2 Big SQL through:
  - Java SQL Shell (JSqsh).
  - Db2 Big SQL console.
  - Tools that support the IBM JDBC/ODBC driver.

## Review questions (cont.)

4. What is the feature that Db2 Big SQL uses to query data efficiently on Hadoop and combine data that is spread across different enterprise data warehouses?
  - A. Self-tuning memory manager (STMM).
  - B. Query optimizer.
  - C. Federation capability.
  - D. Workload manager (WLM).
5. Which of the following items is responsible for communicating with the Hive metastore to gather metadata, and to bridge the RDBMS domain and the Hadoop domain?
  - A. Db2 Big SQL head.
  - B. SQL engine's query optimizer.
  - C. Db2 Big SQL worker.
  - D. Db2 Big SQL scheduler.



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-24. Review questions (cont.)

Write your answers here:

6. C. Federation capability.
7. D. Db2 Big SQL scheduler.

## Review answers

1. What are the reasons that your organization would use Db2 Big SQL?
  - A. Want to access your Hadoop data without using MapReduce.
  - B. It has superior SQL-on-Hadoop performance to optimize data ingestion and query performance for your enterprise.
  - C. No deep learning curve because it is compliant with industry standards for SQL.
  - D. No need to change the format or migrate the data out of Hadoop to do any work on the data.
  - E. All the above.
2. True or False: For optimal performance and function, the ORC file format is recommended for Db2 Big SQL.
3. True or False: You can access Db2 Big SQL through Db2 Big SQL console only.



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-25. Review answers

## Review answers (cont.)

4. What is the feature that Db2 Big SQL uses to query data efficiently on Hadoop and combine data that is spread across different enterprise data warehouses?
  - A. Self-tuning memory manager (STMM).
  - B. Query optimizer.
  - C. Federation capability.**
  - D. Workload manager (WLM).
5. Which of the following items is responsible for communicating with the Hive metastore to gather metadata, and to bridge the RDBMS domain and the Hadoop domain?
  - A. Db2 Big SQL head.
  - B. SQL engine's query optimizer.
  - C. Db2 Big SQL worker.
  - D. Db2 Big SQL scheduler.**



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-26. Review answers (cont.)

## Exercise: Connecting to the IBM Db2 Big SQL server

Using IBM Db2 Big SQL to access Apache Hadoop Distributed File  
System data

© Copyright IBM Corporation 2021

Figure 1-27. Exercise: Connecting to the IBM Db2 Big SQL server

## Exercise objectives

- In this exercise, you connect to the Db2 Big SQL server by using various techniques. First, you explore the lab environment. Then, you learn how to set up the Java™ SQL shell (JSqsh) and use it to connect to the Db2 Big SQL server. Finally, you explore the Db2 Big SQL service by using the Db2 Big SQL console graphical web interface.
- After completing this exercise, you will be able to:
  - Use the Ambari Web UI to check the status of various services, including Db2 Big SQL.
  - Connect to a Db2 Big SQL server and run Db2 Big SQL queries from the JSqsh shell.
  - Run Db2 Big SQL statements by using the JSqsh shell.
  - Launch the Db2 Big SQL console from the Ambari Web UI.
  - Run SQL statements from the Db2 Big SQL console.
  - Display several monitoring capabilities from the Db2 Big SQL console.



Using IBM Db2 Big SQL to access Apache Hadoop Distributed File System data

© Copyright IBM Corporation 2021

Figure 1-28. Exercise objectives

---

# Unit 2. Creating IBM Db2 Big SQL schemas and tables

## Estimated time

01:00

## Overview

In this unit, you learn how to create Db2 Big SQL schemas and tables.

## Unit objectives

- Describe and create Db2 Big SQL schemas and tables
- Identify and list the Db2 Big SQL data types
- Work with various Db2 Big SQL data definition language (DDLs)
- Load data into Db2 Big SQL tables following best practices

## 2.1. Db2 Big SQL tables and schemas

## Db2 Big SQL tables and schemas

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

Figure 2-2. *Db2 Big SQL tables and schemas*

## Topics

-  Db2 Big SQL tables and schemas
  - Data types that are supported by Db2 Big SQL

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

*Figure 2-3. Topics*

## Db2 Big SQL terminologies

- Warehouse:
  - Default directory in the HDFS where the tables are stored.
  - Defaults to /apps/hive/warehouse/.
- Schema:
  - Tables are organized into schemas.
  - Defaults to /apps/hive/warehouse/bigsql.db.
- Table:
  - A directory with zero or more data files.
  - Example: /apps/hive/warehouse/bigsql.db/test1.
  - Tables may be stored anywhere.

Figure 2-4. Db2 Big SQL terminologies

Here are some Db2 Big SQL terminologies:

- A *warehouse* is the default directory in the HDFS in which the tables are stored. The default location is “/apps/hive/warehouse”.
- A *schema* is the directory under the warehouse directory in which the tables are stored. The tables might be organized by schemas, or a default schema can be used. An example is “/apps/hive/warehouse/bigsql.db”.
- A *table* is a directory with zero or more data files. An example of a table directory within HDFS is “/apps/hive/warehouse/bigsql.db/test1”.

**References:**

- Apache Hive warehouse directory:

<https://www.ibm.com/support/pages/changing-permissions-hive-warehouse-directory-iop-425-hadoop-dev>

- Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bi\\_admin\\_big\\_a\\_enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bi_admin_big_a_enable_authorization.html)

- Schemas:

[https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG\\_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0004105.html](https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0004105.html)

- CREATE TABLE (HADOOP) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_a\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_a_crhaoptbl.html)

## Partitioned tables

- A table can be partitioned into one or more columns.
- The partitioning columns are specified when the tables are created.
- Data is stored within one directory for the specified partition.
- Query predicates can be used to eliminate the need to scan every partition. Scan only what is needed.
- Examples:
  - /apps/hive/warehouse/schema.db/tablename/col1=val1
  - /apps/hive/warehouse/schema.db/tablename/col1=val2

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-5. Partitioned tables

A table can be partitioned into one or more columns. You specify the partitioning columns when you create the table. When a table is partitioned, the data is stored within the directory for the specified partition. Query predicates can then be used to eliminate the need to scan every partition and scan only what is needed, which speeds up the query.

For example, if your table was partitioned by col1, there will be two directories on the DFS:

/apps/hive/warehouse/schema.db/tablename/col1=val1

/apps/hive/warehouse/schema.db/tablename/col1=val2

The names of the two directories are, in fact, “col1=val1” and “col1=val2” for each respective partition.

### Reference:

CREATE TABLE (HADOOP) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

## Partitions: A picture says 1,000 words

### Unpartitioned Customers.data

Amy	CA
Bill	NY
Bob	VT
Jane	NY
Susan	CA
Jeff	VT
Austin	CA
Carl	NY

You need to store the state only ONCE for each partition! State is a “virtual column”.

### California Partition

Amy
Susan
Austin

### New York Partition

Bill
Jane
Carl

### Vermont Partition

Bob
Jeff

Show me all New York customers.

Is reading the NY partition faster than the non-partitioned file?

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-6. Partitions: A picture says 1,000 words

## Creating Db2 Big SQL schemas

- Db2 Big SQL tables are organized into schemas.
- A default schema is created that matches your login name.
- The **USE** command can be used to set the default schema (and to create it if it does not exist).
- The **CREATE SCHEMA** command can be used to create a schema.

You can think of a schema as a database.

```
[dataengineer.ibm.com] [bigsq1] 1> use "newschema";
0 rows affected (total: 0.003s)
[dataengineer.ibm.com] [bigsq1] 1> create hadoop table t1 (c1 int);
0 rows affected (total: 3.942s)
[dataengineer.ibm.com] [bigsq1] 1> insert into t1 values (10);
1 row affected (total: 2.596s)
[dataengineer.ibm.com] [bigsq1] 1> select * from t1;
+---+
| c1 |
+---+
| 10 |
+---+
1 row in results(first row: 0.495s; total: 0.498s)
```

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-7. Creating Db2 Big SQL schemas

Schemas are a way to organize tables. The default schema matches your login name. If you want to specify another default schema, you use the **USE** command. If you run the **USE** statement on a schema that does not exist, it is created. After the **USE** command is launched, the specified schema becomes the default.

A new (non-existing) schema is created by running the **USE** command. Then, a table is created with an unqualified table name. (No schema was explicitly specified.) This action results in the table being created under the current default schema. If you had qualified the table, then it would be created in that schema instead of the default schema.

Reminder: When a schema is created, by default a directory is defined in “/apps/hive/warehouse”. The name of the created directory is “*schemaName.db*”, where “*schemaName*” is the name that you specified. So for the schema that is called “*demo*”, by default you should see the directory “/apps/hive/warehouse/*demo.db*”.

It is possible to define a schema and explicitly place the schema directory elsewhere in HDFS besides the Apache Hive warehouse.

**References:**

- USE command:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/bigsq\\_use.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/bigsq_use.html)
- CREATE SCHEMA statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000925.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000925.html)



## Using the web GUI to browse the HDFS

Access by using Apache Ambari

Name	Size	Last Modified	Owner	Group	Permission	Erasure Coding	Encrypted
app-logs	--	2020-08-28 23:44	yarn	hadoop	drwxrwxrwx	No	
apps	--	2020-08-24 19:06	hdfs	hdfs	drwxr-xr-x	No	
ats	--	2020-08-24 15:19	yarn	hadoop	drwxr-xr-x	No	
atsv2	--	2020-08-24 15:20	hdfs	hdfs	drwxr-xr-x	No	
biginights	--	2020-08-25 17:41	hdfs	hdfs	drwxr-xr-x	No	
hdp	--	2020-08-24 15:19	hdfs	hdfs	drwxr-xr-x	No	

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

Figure 2-8. Using the web GUI to browse the HDFS

You can easily use the Apache Ambari **Files View** menu to browse the HDFS and check out the schemas and tables that you create. By default, the directories are under “/apps/hive/warehouse”. You can specify a different directory if you want.

## Creating a Db2 Big SQL table

- A standard `CREATE TABLE` DDL with **extensions**.

```
create hadoop table users
(
    id      int          not null primary key,
    office_id int         null,
    fname   varchar(30)   not null,
    lname   varchar(30)   not null)
row format delimited
  fields terminated by '|'
  stored as textfile;
```

- The `hadoop` keyword creates the table in the HDFS.
- The row format “delimited” and “textfile” formats are the defaults.
- Constraints are not enforced (but useful for query optimization).

*Figure 2-9. Creating a Db2 Big SQL table*

At the top, you see the syntax for creating a Db2 Big SQL table that is called “users” in the default schema (the user ID). If you are familiar with SQL, most of this statement should look familiar to you. The items that are shown in blue are Db2 Big SQL DDL extensions for Hadoop. For example, the `hadoop` keyword indicates that the data is stored on the cluster in the distributed file system and not as a “local” table on the Db2 Big SQL head node only. Other clauses specify the storage format of the data, in this case, a row-based text file with fields that are delimited by a vertical bar (“|”). Other options and clauses are supported. The main point of this example is that it is standard SQL plus some Hadoop-specific extensions.

**References:**

- CREATE TABLE (HADOOP) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

- Working with different data sources:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_analyzingbigdata.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_analyzingbigdata.html)

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html#reference\\_oyh\\_sw4\\_d4](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html#reference_oyh_sw4_d4)

## Results from previous CREATE TABLE . . .

- Data is stored in the following subdirectory of the Apache Hive warehouse:

/apps/hive/warehouse/myid.db/users

- The default schema is user ID, and you can create schemas.
- “Table” is just a subdirectory under schema.db.
- The table’s data are files within the table subdirectory.

- Metadata is collected (Db2 Big SQL and Apache Hive):

- SYSCAT.\* and SYSHADOOP.\* views

- Optionally, use the LOCATION clause of CREATE TABLE to layer Db2 Big SQL schema over existing DFS directory contents:

- Useful if the table contents already are in DFS.
- Avoids the need to LOAD data into Apache Hive.

*Figure 2-10. Results from previous CREATE TABLE . . .*

Running the previous **CREATE TABLE** statement creates a subdirectory in the Apache Hive warehouse directory.

Because a schema name was not explicitly specified, Db2 Big SQL uses the user ID as the default schema name, which in this example was "myid". Schemas are represented in Apache Hive as folders, so the schema folder is "myid.db". A table in Apache Hive is just a subdirectory under the schema folder, so the USERS table is found in ". . . /hive/warehouse/myid.db/users". When data is loaded into this table, the files are stored under this subdirectory.

Optionally, you can use the **LOCATION** clause with the **CREATE TABLE** statement to specify a different directory on HDFS, which enables you to layer your Db2 Big SQL schema over an existing HDFS directory. This feature is especially useful if the table contents are already stored in a directory within HDFS.

**References:**

- Ingestion by adding files to HDFS:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_ingest\\_hdfs.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_ingest_hdfs.html)

- Catalog views:

[https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG\\_11.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0008443.html?pos=2](https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG_11.5.0/com.ibm.db2.luw.sql.ref.doc/doc/r0008443.html?pos=2)

- Roadmap to the catalog views:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011297.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011297.html)

- Catalog views (Hadoop):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_a\\_cataviews.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_a_cataviews.html)

## More about CREATE TABLE

- **HADOOP keyword:**
  - Must be specified unless you enable SYSHADOOP.COMpatibility\_Mode.
- **EXTERNAL keyword:**
  - Indicates that the table is not managed by the database manager.
  - When the table is dropped, the definition is removed, and the data remains unaffected.
- **LOCATION keyword:**
  - Specifies the DFS directory to store the data files.

```
CREATE EXTERNAL HADOOP TABLE T1
(
  C1 INT NOT NULL PRIMARY KEY
  CHECK (C1 > 0),
  C2 VARCHAR(10) NULL,
  ...
)
...
LOCATION
'/user/myusername/tables/user'
```

Figure 2-11. More about CREATE TABLE

You must use the **HADOOP** keyword to create a table for the Hadoop environment. You can omit the keyword if you enable the global session variable “SYSHADOOP.COMpatibility\_Mode”.

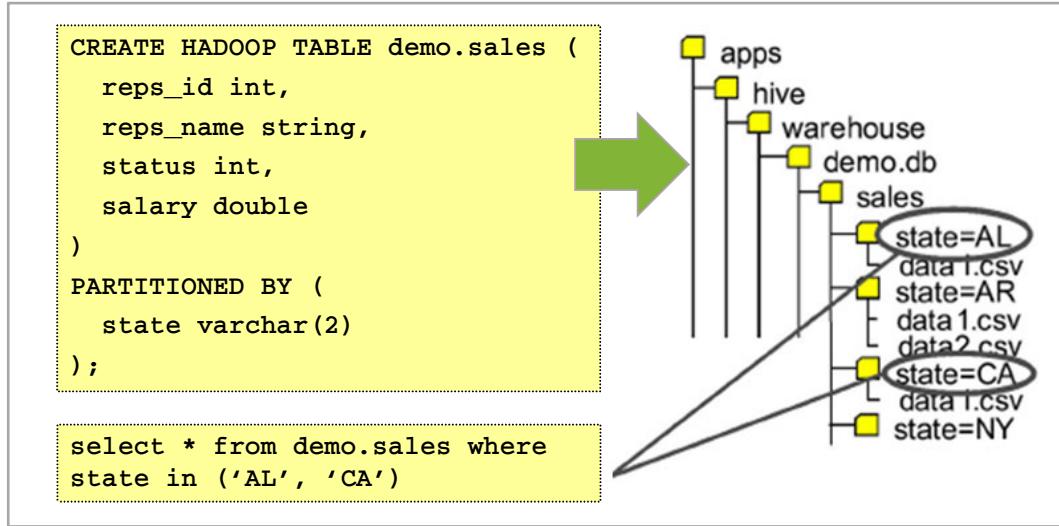
Optionally, you can specify the **EXTERNAL** keyword to create an external table, which indicates that the table is not managed by the database manager. When the table is dropped, the definition of that table is removed, but the data remains untouched. Typically, the **EXTERNAL** keyword is used with the **LOCATION** keyword to specify the directory within the DFS to store the data files.

### References:

- “COMPATIBILITY\_MODE” global variable:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsql.commsql.doc/doc/biga\\_compat.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsql.commsql.doc/doc/biga_compat.html)
- **CREATE TABLE (HADOOP)** statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsql.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsql.commsql.doc/doc/biga_crhaooptbl.html)

## CREATE TABLE: Partitioned tables

- Like Apache Hive, Db2 Big SQL also has partitioned tables.
- Partitioned on one or more columns.
- Query predicates are used to eliminate unwanted data, which speeds up the query.



Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

Figure 2-12. CREATE TABLE: Partitioned tables

Db2 Big SQL has another similarity with Apache Hive: partitioned tables. Db2 Big SQL can create partitioned tables to split up the data into multiple files. The benefit of this action is that the access to the data does not require all the data to be read. The query predicate determines which partition to scan through.

Here is some sample code:

```

CREATE HADOOP TABLE demo.sales (
    reps_id int,
    reps_name string,
    status int,
    salary double
)
PARTITIONED BY (
    state varchar(2)
);

```

In this code, you create a partitioned table on the state column. You do not specify the state column in the original table definition, but in the **PARTITIONED BY** clause. When you run the query to select data from a particular state or states, only the partitions that are specified in the query are retrieved. In the following example, only the partitions where “state=AL” and “state=CA” are retrieved:

```
select * from demo.sales where state in ('AL', 'CA')
```

**Reference:**

**CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/biga_crhaooptbl.html)

## More CREATE TABLE features

- Constraints can be defined inline in the table definition.
- Various file formats are available for **STORED AS**:
  - Text
  - Sequence
  - Optimized Row Columnar (ORC)
  - Parquet
  - More!
- **NULL DEFINED AS** clause for **ROW FORMAT DELIMITED**:
  - Explicit syntax for defining a null value in a delimited file
- Support for **CREATE TABLE LIKE** to clone another table.

*Figure 2-13. More CREATE TABLE features*

The **CREATE TABLE (HADOOP)** statement defines a Db2 Big SQL table that is based on an Apache Hive table for the Hadoop environment. The definition must include its name and the names and attributes of its columns. The definition can include other attributes of the table, such as its primary key or check constraints, which can be defined inline in the table definition.

- PRIMARY KEY
- UNIQUE
- FOREIGN KEY
- CHECK
- REFERENCES

Constraints are only advisory for Hadoop tables, or applied on read, and not enforced.

However, the query optimizer can leverage these constraints to create better query plans. So, even though they are not enforced, they are useful for query optimization.

There are also table types that are available for the **STORED AS** clause. Db2 Big SQL supports a text sequence file, Apache Hive ORC files, Parquet files, and a few others.

The **NULL DEFINED AS** syntax explicitly declares how a null value is represented. The default is a literal backslash character followed by an uppercase N (\N).

There is support for **CREATE TABLE LIKE** to clone another table.

### References:

- CREATE TABLE (HADOOP) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaoptbl.html)

- Performance capabilities (query optimizer):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview\\_performance.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview_performance.html)

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html#reference\\_oyh\\_sw4\\_d4](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html#reference_oyh_sw4_d4)

## CREATE VIEW

- The **CREATE VIEW** statement defines a view on one or more tables, views, or nicknames.
- Uses standard SQL syntax.

```
create view my_users as  
select fname, lname from bigsql.users where id > 100;
```

Figure 2-14. CREATE VIEW

You can create Db2 Big SQL views like you can create a view in an RDBMS. The slide shows a simple example.

### Reference:

#### **CREATE VIEW** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000935.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000935.html)

## Loading data into Db2 Big SQL tables

- Populating tables by using **LOAD**:
  - Best runtime performance
- Populating tables by using **INSERT**:
  - **INSERT INTO ... SELECT FROM**  
Parallel read and write operations
  - **INSERT INTO ... VALUES (...)**  
*Not parallelized. One file per insert. Not recommended except for quick tests.*
- Populate tables by using **CREATE TABLE... AS SELECT...**:
  - Create a Db2 Big SQL table based on the contents of other tables.

Figure 2-15. Loading data into Db2 Big SQL tables

There are two ways to get your HDFS data into Db2 Big SQL tables. Later, you see how to create Db2 Big SQL tables on top of your Hadoop data, but for now here are three ways to load data into these tables:

- The **LOAD** operation is the recommended operation because it yields the best performance.
- You can use **INSERT** operations. There are two types of **INSERT** operations:
  - **INSERT INTO ... SELECT FROM** performs the read and write operations in parallel.
  - **INSERT INTO ... VALUES (...)** is *not* parallelized and produces one file per insert, which is a *highly inefficient* for your Hadoop data. Do not use this operation unless you are testing simple and quick operations.
- Use the **CREATE TABLE ... AS SELECT ...** operation during table creation.

**References:**

- **LOAD HADOOP** statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_load\\_from.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_load_from.html)
- Ingestion with **LOAD HADOOP**:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_ingest\\_load.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_ingest_load.html)
- **INSERT** statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html)
- Parallel **INSERT** in column-organized tables:  
[https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG\\_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0070268.html](https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0070268.html)
- **CREATE TABLE** statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000927.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000927.html)

## Populating Db2 Big SQL tables by using LOAD

- Typically, the best runtime performance.
- Load data from local or remote file systems.

```
load hadoop using file url
'ftp://myID:myPassword@myServer.ibm.com:22/install-
dir/bigsq1/samples/data/GOSALES DW.GO_REGION_DIM.txt' with SOURCE PROPERTIES
('field.delimiter'='\t')    INTO TABLE gosalesdw.GO_REGION_DIM overwrite;
```

- Load data from RDBMS (Db2, Netezza, Teradata, Oracle, MS-SQL, or Informix) by using a JDBC connection.

```
load hadoop
using jdbc connection url 'jdbc:db2://some.host.com:portNum/sampledb'
with parameters (user='myID', password='myPassword')
from table MEDIA columns (ID, NAME)
where 'CONTACTDATE < ''2012-02-01'''
into table media_db2table_jan overwrite
with load properties ('num.map.tasks' = 10);
```

*Figure 2-16. Populating Db2 Big SQL tables by using LOAD*

After you create a Db2 Big SQL table, you can use the **LOAD** command to populate it with data from files in a remote file system, files in your distributed file system, or data in the remote RDBMS servers that are listed. For RDBMS data, you specify JDBC URL properties and either enter an SQL query or a table name to identify the data to be retrieved. Behind the scenes, **LOAD** uses Sqoop connectors to retrieve the necessary data from the source.

In general, **LOAD** typically offers the best runtime performance for populating tables with data. However, there are some other options that you might want to be aware of.

### Reference:

#### LOAD HADOOP statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq1.commsql.doc/doc/big\\_a\\_load\\_from.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq1.commsql.doc/doc/big_a_load_from.html)

## Populating Db2 Big SQL tables by using INSERT (1 of 2)

**INSERT INTO ... SELECT FROM ...**

```
CREATE HADOOP TABLE IF NOT EXISTS big_sales_parquet
( product_key INT NOT NULL, product_name VARCHAR(150) ,
Quantity INT, order_method_en VARCHAR(90) )
STORED AS parquetfile;

-- source tables do not need to be in Parquet format
insert into big_sales_parquet
SELECT sales.product_key, pnumb.product_name, sales.quantity,
meth.order_method_en
FROM sls_sales_fact sales, sls_product_dim prod,sls_product_lookup pnumb,
sls_order_method_dim meth
WHERE
pnumb.product_language='EN'
AND sales.product_key=prod.product_key
AND prod.product_number=pnumb.product_number
AND meth.order_method_key=sales.order_method_key
and sales.quantity > 5500;
```

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

Figure 2-17. Populating Db2 Big SQL tables by using INSERT (1 of 2)

In some cases, it might be more convenient to populate a table with data by using an **INSERT** statement. The **INSERT INTO...SELECT FROM** statement supports parallel read and write operations and can be a convenient way to populate a table with data that is retrieved from a query, particularly if you want to change the underlying storage format that is used for data.

### References:

- **INSERT** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html)

- Parallel **INSERT** in column-organized tables:

[https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG\\_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0070268.html](https://www.ibm.com/support/producthub/db2/docs/content/SSEPGG_11.5.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0070268.html)

## Populating Db2 Big SQL tables by using INSERT (2 of 2)

**INSERT INTO ... VALUES (...):**

- Not parallelized.
- One file is created per insert statement.
- Not recommended except for testing.

```
Create table foo (col1 int, col2 varchar(10));
insert into foo values (1, 'hello');
```

Figure 2-18. Populating Db2 Big SQL tables by using INSERT (2 of 2)

The traditional **INSERT INTO...VALUES(...)** format is also supported but not recommended for anything but simple test operations. Its work is never parallelized, and each **INSERT** results in a separate file.

## Populating Db2 Big SQL tables by using CREATE TABLE ... AS SELECT ...

Source tables can be in different file formats or use a different underlying storage mechanism.

```
-- source tables in this example are external (just DFS files)
CREATE HADOOP TABLE IF NOT EXISTS sls_product_flat
( product_key INT NOT NULL
, product_line_code INT NOT NULL
, product_type_key INT NOT NULL
, product_type_code INT NOT NULL
, product_line_en VARCHAR(90)
, product_line_de VARCHAR(90)
)
as select product_key, d.product_line_code, product_type_key,
product_type_code, product_line_en, product_line_de
from extern.sls_product_dim d, extern.sls_product_line_lookup l
where d.product_line_code = l.product_line_code;
```

*Figure 2-19. Populating Db2 Big SQL tables by using CREATE TABLE ... AS SELECT ...*

Another way to load data into Db2 Big SQL table is to do so along with table creation. The source tables can be in different file formats or use a different underlying storage mechanism.

### Reference:

#### **CREATE TABLE** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/doc/r0000927.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/doc/r0000927.html)

## 2.2. Data types that are supported by Db2 Big SQL

## Data types that are supported by Db2 Big SQL

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

*Figure 2-20. Data types that are supported by Db2 Big SQL*

## Topics

- Db2 Big SQL tables and schemas
- ▶ Data types that are supported by Db2 Big SQL

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

*Figure 2-21. Topics*

## Data types

- Db2 Big SQL uses HCatalog (Apache Hive Metastore) as its underlying data representation and access method.
- SQL is the data type that the database engine supports.
- Apache Hive type:
  - This data type is defined in the Apache Hive Metastore for the table.
  - This type tells SerDe how to encode and decode values for the type.
  - The Db2 Big SQL reader converts values in the Apache Hive types to SQL values on read.

[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-22. Data types

Db2 Big SQL uses HCatalog (and thus the Apache Hive Metastore) as its underlying data representation and access method. Therefore, there are several important distinctions about Db2 Big SQL data types that must be understood.

Some data types that are provided by Db2 Big SQL are data types that are not available in Apache Hive. However, when physically stored or retrieved from the underlying data source, these types are always represented as a data type that Apache Hive does understand. These data types are referred to as *extended data types*.

In some cases, the Apache Hive type is mapped to the nearest Db2 Big SQL data type. For example, the Apache Hive TINYINT data type is supported. However, it is promoted to a SMALLINT upon retrieval from Apache Hive and treated as a SMALLINT throughout the SQL statement that accesses the column.

**References:**

- Data types in Db2 Big SQL:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bsql\\_data\\_types.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bsql_data_types.html)
- Developing Db2 Big SQL applications in your Hadoop environment:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bsql\\_reference.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bsql_reference.html)

## Data types (cont.)

- Various primitive types are supported:
  - TINYINT, INT, DECIMAL, FLOAT, REAL, CHAR, VARCHAR, TIMESTAMP, DATE, VARBINARY, and BINARY.
- Complex types:
  - ARRAY: Ordered collection of elements of the same data type.
  - Associative ARRAY (equivalent to Apache Hive MAP type): Has no specific upper bound on the number of elements. Each element is referenced by its associated index value.
  - ROW (equivalent to Apache Hive STRUCT type): Collection of elements of different types.
  - JSON data can contain nested data in both array-of-rows and map-of-rows contexts.
  - The elements in an ARRAY or ROW can be compared only to another expression of a primitive type.

```
CREATE HADOOP TABLE mytable (id INT, info INT ARRAY[10]);
SELECT * FROM mytable WHERE info[8]=12;
```

*Figure 2-23. Data types (cont.)*

Db2 Big SQL supports many popular data types, including various primitive types that are common to RDBMSs. A subset is listed in the slide. Large objects (LOBs) are supported, but the LOB values are subject to restrictions. Certain complex types are also supported, including arrays and structured (or row) types.

**References:**

- Large objects (LOBs):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008473.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008473.html)
- Array values:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0055424.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0055424.html)
- Row values:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_RowValue.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_RowValue.html)
- Db2 Big SQL complex data types and JSON data:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_complex\\_json.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_complex_json.html)
- **CREATE TABLE (HADOOP) statement:**  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaoptbl.html)

## Data type mapping

CREATE TABLE (HADOOP) data type	Apache Hive data type	SQL data type
TINYINT	TINYINT	SMALLINT
SMALLINT	SMALLINT	SMALLINT
INT	INT	INT
BIGINT	BIGINT	BIGINT
DECIMAL	DECIMAL (introduced in Apache Hive 0.11.0 with a precision of 38 digits)	DECIMAL
REAL	FLOAT	REAL
FLOAT	DOUBLE	FLOAT
DOUBLE	DOUBLE	DOUBLE
CHAR	CHAR	CHAR
VARCHAR	VARCHAR	VARCHAR
STRING	STRING	VARCHAR(32672)
TIMESTAMP	TIMESTAMP	TIMESTAMP(9)
TIMESTAMP(n)	TIMESTAMP	TIMESTAMP(n)
DATE	DATE	DATE
BOOLEAN	BOOLEAN	BOOLEAN © Copyright IBM Corporation 2021

Figure 2-24. Data type mapping

The table in the slide shows the data type mappings between types that are supported in the `CREATE TABLE (HADOOP)` syntax, the corresponding types in the Apache Hive catalog, and the native SQL types that Db2 Big SQL uses at run time.

When you use `DATE STORED AS DATE`, the Apache Hive representation is `DATE`.

As an example, when you create a table, Apache Hive has a type that is called `STRING`. The Db2 Big SQL run time does not have a notion of `STRING`. It cannot support arbitrary string, so when it is defined in the catalog, it is defined as `VARCHAR`. In other words, when you create a table and specify a string column, the Db2 Big SQL engine creates that column as a `VARCHAR` of 32 KB.

### References:

- **CREATE TABLE (HADOOP) statement (data type mappings):**  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)
- Data types that are supported by Db2 Big SQL for Hadoop and HBase tables:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_numbers.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_numbers.html)

## REAL and FLOAT types

- REAL is a 32-bit floating point value.
- FLOAT is a synonym for DOUBLE.
- Apache Hive FLOAT → Db2 Big SQL REAL.

### Apache Hive

```
CREATE TABLE T1
(
    C1 FLOAT
)
```

### Db2 Big SQL

```
CREATE HADOOP TABLE T1
(
    C1 REAL
)
```



[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-25. REAL and FLOAT types

Db2 Big SQL defines a REAL data type as a 32-bit floating point value. The definition of a FLOAT data type is a synonym for DOUBLE.

In Apache Hive, a FLOAT always refers to a 32-bit floating point value, which corresponds to the JAVA FLOAT data type. In Db2 Big SQL, the REAL data type is a synonym for the Apache Hive FLOAT data type.

To migrate Apache Hive and Db2 Big SQL applications, complete the following actions:

1. For **CREATE HADOOP TABLE** statements, change all columns that are defined as FLOAT to REAL to retain the same storage characteristics.
2. When you access Apache Hive or Db2 Big SQL tables, FLOAT values are treated as DOUBLE values when read, which can produce minor rounding differences.

### Reference:

Data types that are supported by Db2 Big SQL for Hadoop and HBase tables:

[https://www.ibm.com/support/knowledgecenter/bg/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/big\\_numbers.html](https://www.ibm.com/support/knowledgecenter/bg/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/big_numbers.html)

## STRING type

- The STRING data type is mapped directly to Apache Hive STRING.
- By default, STRING becomes VARCHAR 32K.
- *Avoid* using STRING:
  - It can cause significant performance degradation.
  - If a value in the underlying storage exceeds the maximum VARCHAR length, that value is silently truncated when converted to the Db2 Big SQL native data type.
  - If the underlying data does not require the maximum VARCHAR length for storage, Db2 Big SQL allocates unnecessary resources for the handling of that column.
- Some alternatives:
  - The best option is to use a VARCHAR that matches your actual needs.
  - The `bigsq1.string.size` property can be used to adjust the size of the VARCHAR that is used to represent the STRING type.
  - A property can be set globally by updating the `$BIGSQL_HOME/conf/bigsq1-conf.xml` configuration file.

```
[localhost][bigsq1] 1> set hadoop property 'bigsq1.string.size'=4096;
[localhost][bigsq1] 1> create hadoop table t1 (fname string, lname, string);
```

Figure 2-26. STRING type

Db2 Big SQL does not have this notion of a STRING, so when it sees a STRING type in the **CREATE** command, it converts it into VARCHAR 32K, and this situation has several implications:

- If a value in the underlying storage exceeds the maximum VARCHAR length, that value is silently truncated when converted to the Db2 Big SQL native data type.
- If the underlying data does not require the maximum VARCHAR length for storage (for example, if the column never exceeds 100 characters), then Db2 Big SQL allocates unnecessary resources for the handling of that column. To improve performance, use a VARCHAR(*n*) with a defined size instead of a STRING. Alter the table to define an appropriate length for the column by using the following syntax:

```
ALTER TABLE <schema>.<table> ALTER COLUMN <col> SET DATA TYPE VARCHAR(<size>)
```

This **ALTER** operation changes the definition of the table both in Db2 Big SQL and in Apache Hive. If data in the VARCHAR(*n*) column exceeds <size>, a null value is returned when that data is queried.

The default behavior for the STRING data type is to map the type to the SQL data type of VARCHAR(32762). The default behavior can lead to performance issues. You can use the configuration property “bigsq.string.size” to change the size of the VARCHAR that is used to represent the STRING type. You can set this property in one of two ways:

- Run the **SET HADOOP PROPERTY** command.
- Set the value globally by updating the “\$BIGSQL\_HOME/conf/bigsq-conf.xml configuration” file.

### References:

- Data types that are supported by Db2 Big SQL for Hadoop and HBase tables:  
[https://www.ibm.com/support/knowledgecenter/bg/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_numbers.html](https://www.ibm.com/support/knowledgecenter/bg/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_numbers.html)
- **CREATE TABLE (HADOOP)** statement:  
[https://www.ibm.com/support/knowledgecenter/bg/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/bg/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

## Unit summary

- Describe and create Db2 Big SQL schemas and tables
- Identify and list the Db2 Big SQL data types
- Work with various Db2 Big SQL data definition language (DDLs)
- Load data into Db2 Big SQL tables following best practices

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

*Figure 2-27. Unit summary*

## Review questions

1. True or False: The `INSERT` operation is the recommended method for getting data into your Db2 Big SQL table for best performance.
  
2. What does the `EXTERNAL` keyword indicate, and when it is added to the `CREATE TABLE` statement?
  - A. The table is not managed by the database manager.
  - B. When the table is dropped, the definition of that table is removed, but the data remains untouched.
  - C. It specifies the directory within the DFS to store the data files.
  - D. A and B.
  
3. True or False: Using the default `STRING` data type causes performance degradation.



[Creating IBM Db2 Big SQL schemas and tables](#)

© Copyright IBM Corporation 2021

Figure 2-28. Review questions

Write your answers here:

1. False, `LOAD` operation is the recommended method for getting data into your Db2 Big SQL table for best performance.
2. D. A and B
3. True, when you create a table, and specify a string column, the Db2 Big SQL engine creates that column as a `VARCHAR` of 32KB, which can lead to performance degradation. It is recommended to use the `VARCHAR` that you need or change the default size.

## Review questions (cont.)

4. Which of the following data types does Db2 Big SQL support?
  - A. TINYINT.
  - B. CHAR.
  - C. TIMESTAMP.
  - D. ARRAY.
  - E. All the above.
  
5. True or False: The `PARTITIONED BY` clause creates a directory in the DFS that contains data for each unique value in the column that is being partitioned, which enhances performance for the query's search criteria.



Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

Figure 2-29. Review questions (cont.)

Write your answers here:

4. E. All the above
5. True

## Review answers

1. True or False: The `INSERT` operation is the recommended method for getting data into your Db2 Big SQL table for best performance.
2. What does the `EXTERNAL` keyword indicate, and when it is added to the `CREATE TABLE` statement?
  - A. The table is not managed by the database manager.
  - B. When the table is dropped, the definition of that table is removed, but the data remains untouched.
  - C. It specifies the directory within the DFS to store the data files.
  - D. A and B.
3. True or False: Using the default `STRING` data type causes performance degradation.



## Review answers (cont.)

4. Which of the following data types does Db2 Big SQL support?
  - A. TINYINT.
  - B. CHAR.
  - C. TIMESTAMP.
  - D. ARRAY.
  - E. All the above.
  
5. **True** or False: The PARTITIONED BY clause creates a directory in the DFS that contains data for each unique value in the column that is being partitioned, which enhances performance for the query's search criteria.



Figure 2-31. Review answers (cont.)

## Exercise: Creating and managing Db2 Big SQL schemas and tables

Creating IBM Db2 Big SQL schemas and tables

© Copyright IBM Corporation 2021

*Figure 2-32. Exercise: Creating and managing Db2 Big SQL schemas and tables*

## Exercise objectives



- In this exercise, you start by creating and dropping a simple Hadoop table by using Db2 Big SQL. Then, you create multiple tables by using various data types and load the tables with data. You also work with views, external tables, and other methods of creating Db2 Big SQL tables.
- After completing this exercise, you will be able to:
  - Work with Db2 Big SQL sample data and create sample tables.
  - Copy sample data into DHFS.
  - Load sample data into tables by using Db2 Big SQL.
  - Create views that span multiple tables.
  - Query views.
  - Populate a table with data based on the results of a query by using Db2 Big SQL.
  - Create and work with external tables.

---

# Unit 3. File formats and querying IBM Db2 Big SQL tables

## Estimated time

01:00

## Overview

In this unit, you learn about file formats and querying Db2 Big SQL tables.

## Unit objectives

- Describe the file formats that are supported by Db2 Big SQL.
- Query Db2 Big SQL tables by using various Data Manipulation Languages (DMLs).

## 3.1. File formats and Db2 Big SQL

## File formats and Db2 Big SQL

[File formats and querying IBM Db2 Big SQL tables](#)

© Copyright IBM Corporation 2021

*Figure 3-2. File formats and Db2 Big SQL*

## Topics

-  File formats and Db2 Big SQL
  - Querying Db2 Big SQL tables

[File formats and querying IBM Db2 Big SQL tables](#)

© Copyright IBM Corporation 2021

*Figure 3-3. Topics*

## File formats and Db2 Big SQL

- Generally, Db2 Big SQL supports any Hadoop format.
- Common formats (STORED AS . . . ):
  - Text
  - Sequence (binary and text sequence files)
  - Parquet
  - Optimized Row Columnar (ORC)
  - Record Columnar (RC)
  - Avro

[File formats and querying IBM Db2 Big SQL tables](#)

© Copyright IBM Corporation 2021

Figure 3-4. File formats and Db2 Big SQL

The Hadoop environment supports many file formats. The formats that are described here are available either by using explicit SQL syntax, such as “STORED AS ORC”, or by using installed interfaces such as Avro. Columnar storage saves both time and space during big data processing. The ORC and Parquet file formats provide excellent performance advantages when used with Db2 Big SQL.

Here are the Db2 Big SQL file formats that are covered in detail in the upcoming slides:

- Text
- Sequence:
  - Binary sequence file
  - Text sequence file
- Parquet
- Optimized Row Columnar (ORC)
- Record Columnar (RC)
- Avro

**Reference:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
a\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Text (1 of 5)

```
create hadoop table users (
    user_id int,
    fname   varchar(10),
    hire    date
)
row format delimited
    fields terminated by '|'
```

The text file format is the default storage format for a table. The underlying data is stored in delimited form with one record per line and new line characters separating individual records. An example is a comma-separated value (CSV) file.

**Pros:**

- Supported by the I/O engine.
- Human-readable.
- Supported by most tools.

**Cons:**

- Least efficient file format.
- Data conversion to binary format is costly.

Figure 3-5. Text (1 of 5)

The text file format is the default storage format for a table. The underlying data is stored in delimited form with one record per line and new line characters separating individual records.

The text file format requires type conversion for all non-character columns, so it is not as efficient as a binary storage format.

This file format does not support compression.

You can specify delimiters by using the **ROW FORMAT DELIMITED** clause in the **CREATE TABLE (HADOOP)** statement. For example:

```
create hadoop table users (
    user_id int,
    fname   varchar(10),
    hire    date
)
row format delimited
    fields terminated by '|'
```

**Advantages:**

- Supported by the I/O engine.
- Human readable.
- Supported by most tools.

**Disadvantages:**

- Least efficient file format.
- Data conversion to binary format is costly.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_crhaooptbl.html)

- Db2 Big SQL readers and writers:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_iibraries.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_iibraries.html)

## Text (2 of 5)

- **TERMINATED BY** indicates delimiters:
  - Default delimiter: ASCII 0x01 (Ctrl+a).
  - Delimiters can be specified as characters (for example, '|').
  - Common escape sequences (for example, '\t').
- Your data must not contain the delimiter character.
  - It is possible to escape the character if you specify the **ESCAPED BY** clause.
  - Newline characters cannot be escaped.
  - Use '\\\' to specify a literal backslash.
- **NUL DEFINED AS** indicates how a literal **NULL** is represented.

```
CREATE HADOOP TABLE texttable (
  user_id INT NOT NULL,
  name VARCHAR(10)
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  LINES TERMINATED BY '\n'
  NULL DEFINED AS '\N'
STORED AS TEXTFILE;
```

Figure 3-6. Text (2 of 5)

You can specify delimiters by using the **ROW FORMAT DELIMITED** clause in the **CREATE TABLE (HADOOP)** statement, as shown in the example on the slide.

The individual values are separated by the field terminator (ASCII character *0x01*, by default). However, you can specify values with the **FIELDS TERMINATED BY** clause.

Generally, your data must not contain delimiter characters. The only exception is that field terminator characters can exist within your data if you specify the **ESCAPED BY** clause and these characters are properly escaped. Newline characters cannot be escaped.

If an incompatible column value is provided (for example, you attempt to insert the value "a" into a column that is defined as INT), that value is treated as a null value. If null values are not allowed, an error is returned. If your input data contains more columns (fields) than the table, the additional fields in the data are ignored. If there are fewer fields in the input data than columns in the table, null values are inserted if the columns are defined as nullable; otherwise, an error is returned.

When the row key consists of more than one column, the following rules apply:

- **NUL DEFINED AS** values are represented by the value that you specify in the **NUL DEFINED AS** clause (or "\N" by default).
- At least one column must have a non-null value.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaoptbl.html)

- HBase:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_hbase\\_workingwith.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_hbase_workingwith.html)

## Text (3 of 5)

The following table details the allowable formats for Hive data types.

Type	Format	Type	Format
<b>TINYINT</b>	0,110,-4	<b>BOOLEAN</b>	0,1 (Note: 0=FALSE, 1=TRUE)
<b>FLOAT, DOUBLE</b>	0.0,0.1831,9.81,3.12E-44	<b>TIMESTAMP</b>	YYYY-MM-DD hh:mm:ss.f+ (Ex:1997-11-03 19:22:58.13)
<b>STRING, CHAR</b>	any	<b>DATE</b>	YYYY-MM-DD (Ex: 2011-07-14)

Figure 3-7. Text (3 of 5)

This table uses the Hive data type, where it shows formats of STRING encoded data types. Remember, some Db2 Big SQL data types are represented as different Hive data types. For example, the Big SQL REAL data type is stored as a Hive FLOAT data type.

### Reference:

HBase (the allowable formats for Hive data types):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/big\\_a\\_hbase\\_workingwith.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/big_a_hbase_workingwith.html)

## Text (4 of 5)

The incompatible column value is provided. It is treated as NULL on read. If null values are not allowed, an error is returned.

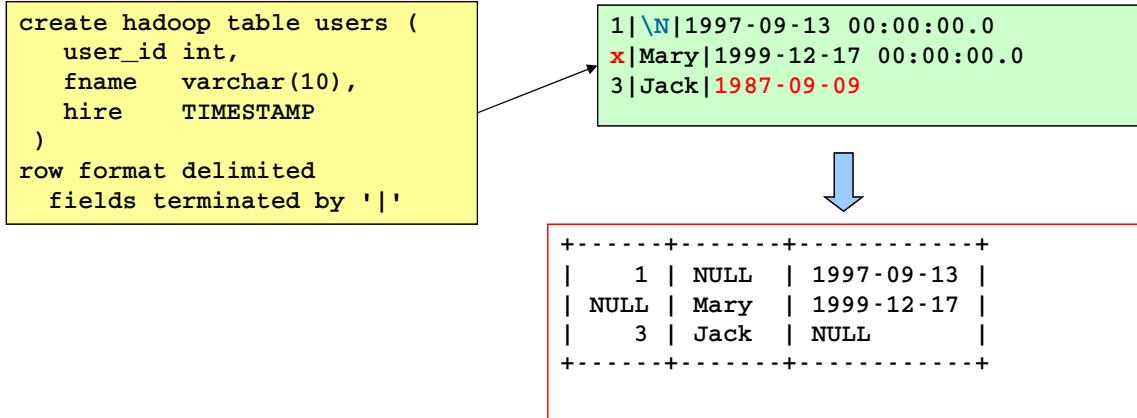


Figure 3-8. Text (4 of 5)

If an incompatible column value is provided (for example, you attempt to insert the value “a” into a column that is defined as INT), that value is treated as a null value. If null values are not allowed, an error is returned.

In this example, you see that a table has three columns:

- A *user\_id* of type int
- A *fname* of type varchar(10)
- A *hire* of type TIMESTAMP

Then, in the data file, you have three rows.

- The first row contains the values of type “int”, “\N”, and the timestamp. They are added correctly because the data format matches the columns’ data types.
- The second row, in the first column, is expecting an “int”, but it gets a “char”, so when you insert the data, you get a NULL. If the column was defined with a NOT NULL, the query fails.
- In the third row, the last column’s date is just a date without an extra time value. It is inserted as a NULL because Hive expects a TIMESTAMP for the DATE column.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

## Text (5 of 5)

- Generally, your actual data must not contain delimiter characters.

1,Hello,World!,3

This example is treated as four column values, not three.

- The only exception is that field terminator characters can exist within your data if you specify the **ESCAPED BY** clause and these characters are properly escaped.

```
create hadoop table messages
(
    msg_id  int,
    msg      varchar(200),
    msg_sev int
)
row format delimited
    fields terminated by ',' escaped by '\\'
```

1>Hello\, World!, 3



+	-----	+
	1   Hello, World!  3	
+	-----	+

Figure 3-9. Text (5 of 5)

Here you see how to escape certain characters if they are being used as a delimiter. Generally, your actual data must not contain delimiter characters. The only exception is that field terminator characters can exist within your data if you specify the **ESCAPED BY** clause and these characters are properly escaped. Newline characters cannot be escaped.

For example, if you needed to insert the value “1, Hello, World!, 3 “, it is treated as four columns because of the three commas that separate the data. To do this task successfully, you must specify an escape character by using the **ESCAPED BY** clause.

**Example:**

```
create hadoop table messages
(
    msg_id  int,
    msg      varchar(200),
    msg_sev int
)
row format delimited
    fields terminated by ',' escaped by '\\'
```

So, assuming that your escape character is the backslash, you insert the row as:

```
1, Hello\, World!, 3
```

This format correctly inserts the rows into the three columns of that table.

**Reference:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Sequence (1 of 2)

- The sequence file format is used to hold arbitrary data that might not otherwise be “splittable”.
- Track record boundaries to produce splits.
- Useful for storing data that is tough to split otherwise (such as XML or JSON).

```
CREATE HADOOP TABLE text_seq
  (col1 int, col2 varchar(20))
STORED AS SEQUENCEFILE
```

### Pros:

- Supported by the native I/O engine when storing delimited data only.
- Supported by the Java I/O engine for other storage formats.
- Sequence files are relatively common in Hadoop.

### Cons:

- One of the slower storage formats.
- Not human-readable.

Figure 3-10. Sequence (1 of 2)

The sequence file format is used to hold arbitrary data that might not otherwise be “splittable”. For example, in a text file, newline characters (“\n”) are used to determine the boundaries of a record, so a distributed file system (DFS) block can be processed by looking for newline characters.

However, if the data in that file is in binary format or is compressed with an algorithm that does not maintain markers for a record boundary, reading that block is impossible. A sequence file maintains extra metadata to recognize record boundaries.

Some of the pros of using sequence files are that it is supported by the native I/O engine when storing delimited data. For other storage formats, it is supported by the Java I/O engine. One of the more commonly used formats in Hadoop.

A couple of cons of sequence files are that they are relatively slow compared to other formats, and they are not human-readable.

The following compression types are recommended:

- BZIP2
- DEFLATE
- GZIP
- SNAPPY

**Reference:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big\\_a\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big_a_fileformats.html)

## Sequence (2 of 2)

There are two types of sequence files:

- A binary sequence file:

- It stores data in a binary format by using the Hive `LazyBinarySerDe`.
- With binary storage, the data requires little conversion processing while being read.

```
CREATE HADOOP TABLE bin_seq
  (col1 int, col2 varchar(20))
STORED AS BINARY SEQUENCEFILE
```

- A text sequence file

It stores delimited data within the sequence file format, which enables the use of compression algorithms on textual data that otherwise would not be splittable.

```
CREATE HADOOP TABLE text_seq
  (col1 int, col2 varchar(20))
STORED AS TEXT SEQUENCEFILE
```

Figure 3-11. Sequence (2 of 2)

There are two types of sequence files:

- A *binary sequence file* stores data in a binary format by using the Hive “`LazyBinarySerDe`”. With binary storage, the data requires little conversion processing while being read.
- A *text sequence file* stores delimited data within the sequence file format, which enables the use of compression algorithms on textual data that otherwise would not be splittable.

### References:

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_crhaooptbl.html)

## Parquet (1 of 4)

- Open source columnar file format.
- Efficient compression and encoding schemes.
- The recommended compression types for this file format are SNAPPY (the default) and GZIP.

```
CREATE HADOOP TABLE parquet
  (col1 int, col2 varchar(20))
STORED AS PARQUETFILE
```

**Pros:**

- Supported by the native I/O engine.
- Highest performance file format.

**Cons:**

- Not good for data interchange outside of Hadoop.
- Has limited support for DATE or TIMESTAMP data types.

Figure 3-12. Parquet (1 of 4)

The Parquet file format is an open source columnar storage format for Hadoop that supports efficient compression and encoding schemes.

During load and insert operations, the following values are set as the default values for the Parquet format. You can change these values by using the **SET HADOOP PROPERTY** command before you run the **LOAD HADOOP** statement:

- **SET HADOOP PROPERTY 'dfs.blocksize' = 268435456;**
- **SET HADOOP PROPERTY 'parquet.page.size' = 65536;**
- **SET HADOOP PROPERTY 'parquet.dictionary.page.size' = 65536;**
- **SET HADOOP PROPERTY 'parquet.block.size' = 268435456;**
- **SET HADOOP PROPERTY 'parquet.enable.dictionary' = 'true';**
- **SET HADOOP PROPERTY 'parquet.compression' = 'SNAPPY';**

The recommended compression types for this file format are SNAPPY (the default) and GZIP.

Example:

**CREATE HADOOP TABLE parquet (col1 int, col2 varchar(20)) STORED AS PARQUETFILE;**

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big\\_a\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big_a_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_a\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_a_crhaoptbl.html)

## Creating a table with underlying Parquet files (2 of 4)

- Creating a Parquet table is easy.
- Ways of populating a Parquet table:
  - Using `INSERT` from `SELECT`.

```
CREATE HADOOP TABLE parquet
  (col1 int, col2 varchar(20))
  STORED AS PARQUETFILE
```

```
INSERT INTO new_parquet_table
  SELECT * FROM existing_delimited_table
```

- Using `CREATE TABLE AS SELECT`.

```
CREATE HADOOP TABLE new_parquet_table
  STORED AS PARQUETFILE
  AS SELECT * FROM existing_text_table
```

- `INSERT ... VALUES`.

Do not use the `INSERT ... VALUES` operation on HADOOP tables to load large quantities of data into a table. Use it for testing.

```
INSERT INTO parquet VALUES (1,'a'),(2,'b')
```

*Figure 3-13. Creating a table with underlying Parquet files (2 of 4)*

Here are some examples of how to work with Parquet tables (you saw how to create a Parquet table in the previous slide).

Example:

**`CREATE HADOOP TABLE parquet (col1 int, col2 varchar(20)) STORED AS PARQUETFILE;`**

Essentially, everything is the same as creating a regular table except that you specify the **`STORED AS PARQUETFILE`** clause.

There are three ways to populate a Parquet table:

1. Using an **INSERT** from **SELECT** statement to basically transform an existing delimited table into a Parquet table.

Example:

```
INSERT INTO new_parquet_table SELECT * FROM existing_delimited_table;
```

2. Using **CREATE TABLE AS SELECT**.

Example:

```
CREATE HADOOP TABLE new_parquet_table STORED AS PARQUETFILE AS SELECT *  
FROM existing_text_table;
```

3. Using **INSERT ... VALUES** to test out your queries.

Example:

```
INSERT INTO parquet VALUES (1, 'a'), (2, 'b');
```

---



### Reminder

Do not use the **INSERT ... VALUES** operation on Hadoop tables to load large quantities of data into a table. This operation cannot be run in parallel. Each invocation produces a data file, which negatively impacts both **INSERT** performance and **SELECT** performance. You can use the **INSERT ... VALUES** operation for testing or for populating small dimension tables in which all the rows can be inserted in a single operation.

---

### References:

- **INSERT** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000970.html)

- **CREATE TABLE (HADOOP)** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

## Loading a Parquet table (3 of 4)

```
LOAD HADOOP USING FILE URL
  '/user/bigrsql/data/test.tbl'
WITH SOURCE PROPERTIES ('field.delimiter' = '|')
INTO TABLE parquet APPEND;
```

Best practices for loading Parquet data files:

- It is essential to tune the `num.map.tasks` parameter of **LOAD**.
  - Set the value to at least the number of data nodes in your cluster.
  - Performance can be improved by increasing the number of map tasks that are assigned to the **LOAD**.
- When loading large Parquet data files, The most effective **LOAD** strategy is to copy data from a remote data source to the distributed file system (DFS), and then run the **LOAD HADOOP** statement pointing to that DFS data, which results in better performance.
- The file option of the Db2 Big SQL load can be used to perform data conversion from text to Parquet.

*Figure 3-14. Loading a Parquet table (3 of 4)*

Another alternative to getting data into Parquet tables is by using the **LOAD** operation. There are a few best practices for using **LOAD**.

During **LOAD** processing, the Hadoop MapReduce framework is used, which runs jobs by using multiple mappers and reducer tasks. You can control the number of map tasks that are created during the **LOAD** with the `num.map.tasks` parameter. It is essential to tune this parameter to improve the performance by increasing the number of map tasks that are assigned to the **LOAD**.

To update the default value of `num.map.tasks`, include the **WITH LOAD PROPERTIES** clause in your **LOAD** statement. If your files are part of a file system that is remote to the cluster and not on the DFS, set the value to at least the number of data nodes that are in your cluster. But, be aware of the configuration and connection parameters of your remote system, and the number of slots that are available for map tasks as you modify the `num.map.tasks` parameter.

The `num.map.tasks` property influences the number of files that are created on the DFS after the data is loaded. Block size can affect performance when you load files from DFS. Large source files in the DFS are broken into smaller blocks. Each map task processes one or more blocks. **LOAD** uses as many map tasks as possible, which are limited by the property `num.map.tasks`.

Loading from HDFS has better performance, where the most effective **LOAD** strategy is to copy data from a remote data source to the DFS, and then run the **LOAD HADOOP** statement pointing to that DFS data because moving the large files to DFS first and then running **LOAD** enables the files to be split, which takes advantage of running parallel map tasks.

Loading directly from a remote file system might conserve space in the DFS, but you lose the benefits of **LOAD** processing in parallel.

You can use the file option of the Db2 Big SQL load to perform data conversion from text to Parquet.

#### References:

- **LOAD HADOOP** statement:

[https://www.ibm.com/support/knowledgecenter/en/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...msql.doc/doc/biga\\_load\\_from.html](https://www.ibm.com/support/knowledgecenter/en/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...msql.doc/doc/biga_load_from.html)

- How to improve **LOAD HADOOP** performance:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...doc/doc/bigsq\\_loadperf.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...doc/doc/bigsq_loadperf.html)

## Parquet and compression (4 of 4)

- Parquet files are compressed with SNAPPY by default. The recommended compression types for this file format are SNAPPY (the default) and GZIP
- You can explicitly specify a particular compression.

```
SET HADOOP PROPERTY 'parquet.compression' = 'SNAPPY';
INSERT INTO parquet_snappy SELECT * FROM text_table;
```

- It is okay to populate the same table with multiple algorithms.

```
SET HADOOP PROPERTY parquet.compression = SNAPPY;
INSERT INTO parquet_snappy SELECT * FROM text_table;

SET HADOOP PROPERTY parquet.compression = GZIP;
INSERT INTO parquet_snappy SELECT * FROM text_table;
```

Figure 3-15. Parquet and compression (4 of 4)

Regarding Parquet and compression, by default Parquet files are compressed with SNAPPY. The recommended compression types for this file format are SNAPPY (the default) and GZIP.

Set the **HADOOP PROPERTY** *parquet.compression* flag to the compression algorithm of your choice:

**Example:**

```
SET HADOOP PROPERTY 'parquet.compression' = 'SNAPPY';
INSERT INTO parquet_snappy SELECT * FROM text_table;
```

It is also valid to populate the same table with multiple algorithms. Set the property before each insert or load operation.

**Example:**

```
SET HADOOP PROPERTY 'parquet.compression' = 'SNAPPY';
INSERT INTO parquet_snappy SELECT * FROM text_table;
SET HADOOP PROPERTY parquet.compression = GZIP;
INSERT INTO parquet_snappy SELECT * FROM text_table;
```

**Reference:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big\\_a\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big_a_fileformats.html)

## Optimized Row Columnar (1 of 2)

```
CREATE HADOOP TABLE orc_table
  (col1 int, col2 varchar(20))
  STORED AS ORC
```

- This format stores collections of rows in a columnar format.
- It uses type-specific encoders for each column and divides the file into large stripes.
- It is recommended for optimal performance and functioning.

**Pros:**

- Efficiently retrieves individual columns.
- Efficient compression.

**Cons:**

- Supported by only the Java I/O engine.
- Not good for data interchange outside of Hadoop.
- Db2 Big SQL cannot use some advanced ORC features.

Figure 3-16. Optimized Row Columnar (1 of 2)

The ORC file format provides a highly efficient way to store data. ORC files store collections of rows in a columnar format, which enables parallel processing of row collections across your cluster. As of Db2 Big SQL V5.0.2, the ORC file format is recommended for optimal performance and functions.

The ORC file format uses type-specific encoders for each column and divides the file into large stripes. Each stripe uses indexes that enable the Db2 Big SQL readers to skip sets of rows that do not satisfy filter conditions. A footer contains metadata that includes byte range information for each stripe and type information for the file. The amount of resources that Db2 Big SQL uses to process ORC files is affected by the ORC stripe size. The recommended stripe size for ORC files, which is determined by the `orc.stripe.size` property, is 64 MB. Stripe sizes larger than 256 MB should be avoided.

**Example:**

```
CREATE HADOOP TABLE orc_table (col1 int, col2 varchar(20)) STORED AS ORC;
```

**Pros:**

- ORC files allow for effective retrieval of individual columns.
- Efficient compression.

**Cons:**

- ORC is supported by only the Java I/O engine.
- Not good for any data exchange outside of Hadoop.
- Db2 Big SQL cannot use some of the advance ORC feature.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_crhaooptbl.html)

## ORC and compression (2 of 2)

- ORC defines compression at DDL time.

```
CREATE HADOOP TABLE orc_table (col1 int, col2 varchar(20))
  STORED AS ORC
  TBLPROPERTIES (
    'orc.compress' = 'SNAPPY'
  )
```

- The recommended compression type for this file format is ZLIB (the default).
- Available compression schemes are:
  - NONE
  - ZLIB
  - SNAPPY

*Figure 3-17. ORC and compression (2 of 2)*

ORC defines compression at DDL time. You specify it in the **TBLPROPERTIES** clause. Table properties are used to control behavior of files that are stored as ORC, where it ensures that all users store data with the same options.

The available compression schemes are NONE, ZLIB, or SNAPPY. The recommended compression type for this file format is ZLIB (the default).

For more information about table properties, see <https://orc.apache.org/docs/hive-config.html>.

**Example:**

```
CREATE HADOOP TABLE orc_table (col1 int, col2 varchar(20)) STORED AS ORC
TBLPROPERTIES ( 'orc.compress' = 'SNAPPY' );
```

### References:

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaoptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaoptbl.html)

## Record Columnar

- What is it?

- It uses binary key-value pairs.
- It partitions rows horizontally into row splits and then partitions each row split vertically.

Pros:

- Supported by the native I/O engine.
- Efficiently retrieves individual columns.
- Efficient compression.

Cons:

- Not good for data interchange outside of Hadoop.

- ORC has many advantages over RC, including better compression, better memory utilization, support for all data types, and improved query performance.
- Compression types:
  - BZIP2
  - DEFLATE
  - GZIP
  - SNAPPY

Figure 3-18. Record Columnar

The RC file format uses binary key-value pairs. It partitions rows horizontally into row splits and then partitions each row split vertically. The metadata pertaining to a row split is the key part, and all the actual data in the row split is stored as the value part of a record.

The ORC file format has many advantages over the RC file format, including better compression, better memory utilization, support for all data types (including DATE and DECIMAL), and improved query performance. Therefore, the ORC file format is recommended for optimal performance and functioning.

The following compression types are recommended:

- BZIP2
- DEFLATE
- GZIB
- SNAPPY

Example:

```
CREATE HADOOP TABLE orc_table (col1 int, col2 varchar(20)) STORED AS RCFILE
```

**References:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big\\_a\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big_a_fileformats.html)

## Avro (1 of 5)

- Avro is an Apache open source project.
- It is a compact and fast binary data format.
- It is not required to have code generation to read or write data files.
- It relies on schemas for communicating the structure of the data.

### Pros:

- Supported by the native I/O engine.
- Popular binary data exchange format.
- Table structure can be inferred from existing schema.
- Decent performance.

### Cons

- Not as efficient as Parquet or ORC.
- Not human-readable.

Figure 3-19. Avro (1 of 5)

Avro is an Apache open source project that acts as data serialization system. It provides a convenient way to represent complex data structures within the Hadoop environment. Also, it is a compact and fast binary data format that allows simple integration with dynamic languages.

It is not required to have code generation to read or write data files or to use or implement remote procedure call (RPC) protocols. If there is a need to use statically typed languages, use code generation as an optional optimization.

Avro relies on schemas for communicating the structure of the data (in addition to Avro SerDe in your **CREATE TABLE (HADOOP)** statement) that enable you to read or write Avro data as Db2 Big SQL tables.

Pros:

- It is supported by the native I/O engine.
- It is a popular data exchange format.
- The table structure can be inferred from existing schema.
- It has decent performance.

Cons:

- Not as efficient as Parquet or ORC.
- Not human-readable.

For more information about Apache Avro, see <http://avro.apache.org>.

### References:

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Creating an Avro table with an inline Avro schema (2 of 5)

- An Avro table can be created with an inline Avro schema.

```

CREATE EXTERNAL HADOOP TABLE bs_rev_profit_by_campaign (
    Revenue DOUBLE,
    GrossProfit DOUBLE,
    CompaignName VARCHAR(10)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
    INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION '/user/biadmin/avro/sheet/'
TBLPROPERTIES (
    'avro.schema.literal'='{"type":"record",
    "name": "TUPLE_3",
    "fields":
    [ { "name": "Revenue", "type": [ "null", "double" ] },
    {"doc": " autogenerated from Pig Field Schema"},

    {"name": "GrossProfit", "type": [ "null", "double" ] },
    {"doc": " autogenerated from Pig Field Schema"},

    {"name": "CompaignName", "type": [ "null", "string" ] },
    {"doc": " autogenerated from Pig Field Schema"}
    ]
)
;

```

- A Db2 Big SQL schema can be inferred from the Avro schema:
  - Db2 Big SQL and Hive schema evolution with tables in Avro is primarily driven by SerDe's understanding of the Avro table definition.
  - It is critical that you keep the Db2 Big SQL column definitions and the Avro schema synchronized.

Figure 3-20. Creating an Avro table with an inline Avro schema (2 of 5)

There are two ways to create an Avro table. The first way that is shown here is creating an Avro table with an inline Avro schema. You must specify the Avro schema as parameters to an Avro SerDe, and you also must specify the INPUTFORMAT and the OUTPUTFORMAT. Db2 Big SQL schema can be inferred from the Avro schema.

Support for Db2 Big SQL and Hive schema evolution with tables in the Avro file format is primarily driven by the SerDe's understanding of the Avro table definition. When you change the schema, it is critical that you keep the Db2 Big SQL column definitions and the Avro schema synchronized to ensure that data is written and read with the new Avro SerDe definition.

So, if you have an Avro file format that requires a new column, in Db2 Big SQL, you can use the **ALTER TABLE (HADOOP/HBASE)** statement to add a column:

```
ALTER TABLE...
    ADD COLUMN col3 INT;
```

The table property 'avro.schema.literal' or the HDFS file (containing the JSON schema) that is referenced by 'avro.schema.url' also must be updated. For example:

```
ALTER TABLE...
SET TBLPROPERTIES (
    'avro.schema.literal' =
        '{"type": "record",
         "name": "TUPLE_1",
         "fields": [
             ...
             {
                 "name": "col3", "type": ["null", "int"]
             }
         ]
    }'
);
```

You must specify the full new schema. After this statement runs, you can write data to and select data from the altered table, and Db2 Big SQL handles that data according to the new table layout.

A best practice is to not specify a column list, and the correct details are then automatically extracted from the specified Avro schema. If you do specify a column list for the table, ensure that the number of columns matches the number of columns in the Avro schema. Otherwise, the **CREATE TABLE** statement returns an error.

Here is a detailed example:

```

CREATE EXTERNAL HADOOP TABLE bs_rev_profit_by_compaign (
    Revenue DOUBLE,
    GrossProfit DOUBLE,
    CompaignName VARCHAR(10)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
    INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION '/user/biadmin/avro/sheet/'
TBLPROPERTIES (
    'avro.schema.literal'='{"type": "record",
        "name": "TUPLE_3",
        "fields":
            [ { "name": "Revenue", "type": [ "null", "double" ],
                "doc": " autogenerated from Pig Field Schema" },
            { "name": "GrossProfit", "type": [ "null", "double" ],
                "doc": " autogenerated from Pig Field Schema" },
            { "name": "CompaignName", "type": [ "null", "string" ],
                "doc": " autogenerated from Pig Field Schema" }
            ]
    }
    ')
;

```

### References:

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/big_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/big_crhaooptbl.html)

## Creating an Avro table: external schema (3 of 5)

You can point to an external Avro schema file.

```
CREATE HADOOP TABLE avro_external
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  STORED AS
    INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
    LOCATION '/user/bigsql/avro/data/'
    TBLPROPERTIES (
      'avro.schema.url'='hdfs:///user/bigsql/avro/twitter.avsc'
    );
```

*Figure 3-21. Creating an Avro table: external schema (3 of 5)*

The second method is to create an Avro table by using a file. You still must specify the Avro schema as parameters to an Avro SerDe, in addition to the INPUTFORMAT and the OUTPUTFORMAT.

Example:

```
CREATE HADOOP TABLE avro_external
  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  STORED AS
    INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
    LOCATION '/user/bigsql/avro/data/'
    TBLPROPERTIES (
      avro.schema.url='hdfs:///user/bigsql/avro/twitter.avsc'
    );
```

So, the two ways to create an Avro table show that you can use table properties to include an Avro file format, such as 'avro.schema.url' or 'avro.schema.literal', as shown in the following examples:

```
...TBLPROPERTIES (
    'avro.schema.url' = 'file:///path/to/the/schema/test_serializer.avsc'
)
...
;
...TBLPROPERTIES (
    'avro.schema.literal' =
        '{ "namespace": "com.howdy",
            "name": "some_schema",
            "type": "record",
            "fields": [ { "name": "string1", "type": "string"} ] }'
)
...
;
```

### **Reference:**

File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Avro schema data type mapping (4 of 5)

- The following Avro data types are mapped to Db2 Big SQL data types.

Avro data type	Db2 Big SQL data type
BOOLEAN	BOOLEAN
INT	INT
LONG	BIGINT
FLOAT	DOUBLE

Avro data type	Db2 Big SQL data type
DOUBLE	DOUBLE
STRING	VARCHAR(max)
ENUM	VARCHAR(max)

- The STRING data type can be bad for performance. You can set the `bigsq1.string.size` property to default the STRING mapping to a smaller VARCHAR size.

Figure 3-22. Avro schema data type mapping (4 of 5)

By using an Avro SerDe in your **CREATE TABLE (HADOOP)** statement, you can read or write Avro data as Db2 Big SQL tables. The table on the slide shows the mapping of Avro data types to Db2 Big SQL data types.

Using any other data types returns SQL1666N.

As we mentioned before, the STRING data type has no specific size that is associated with it, so the SQL processor might assume that each value in a STRING column always contains 32 KB of data and performance might be impacted. If performance is a concern, use types that contain a specific length. You can also set the `bigsq1.string.size` property to default the STRING mapping to a smaller VARCHAR size.

Set this property in one of two ways:

- Run the **SET HADOOP PROPERTY** command.
- Set the value globally by updating the `$BIGSQL_HOME/conf/bigsq1-conf.xml` configuration file.

**References:**

- File formats that are supported by Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga\\_fileformats.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/biga_fileformats.html)

- **CREATE TABLE (HADOOP) statement:**

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga\\_crhaooptbl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/biga_crhaooptbl.html)

## Avro schema example (5 of 5)

```
CREATE EXTERNAL HADOOP TABLE avro_external
  (username varchar(20), tweet varchar(140), timestamp bigint)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
  INPUTFORMAT
'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
  OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
  LOCATION '/user/bigsql/avro/data/'
  TBLPROPERTIES (
    'avro.schema.url'='hdfs:///user/bigsql/avro/twitter.avsc'
);
```

Figure 3-23. Avro schema example (5 of 5)

## 3.2. Querying Db2 Big SQL tables

## Querying Db2 Big SQL tables

File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

*Figure 3-24. Querying Db2 Big SQL tables*

## Topics

- File formats and Db2 Big SQL
- ▶ Querying Db2 Big SQL tables

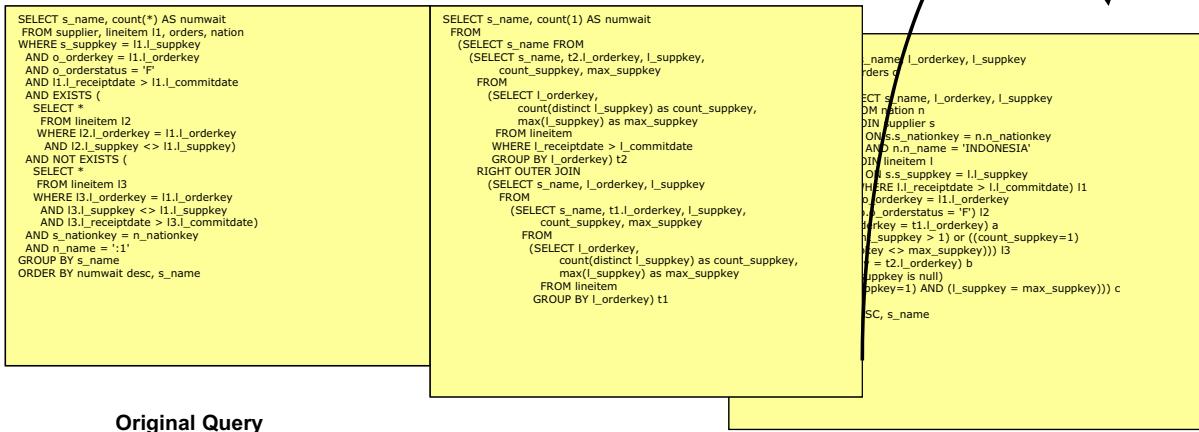
File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

*Figure 3-25. Topics*

## Power of standard SQL

- Everyone loves performance numbers, but they are not the whole story. How much work must you do to achieve those numbers?
- Db2 Big SQL can run all 99 queries in the Transaction Processing Performance Council (TPC)-DS benchmark as is or with allowed minor query modifications.



File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

Figure 3-26. Power of standard SQL

Some of the most popular query workloads were defined by the Transaction Processing Performance Council (TPC). These workloads were used for years for performance benchmarking of RDBMSs. For more information about TPC, see <http://www.tpc.org/>.

Db2 Big SQL adheres to standard SQL syntax, allowing it to handle large, complex queries and allowing you to use system-generated complex queries. The 99 queries in the TPC-DS benchmark contain various SQL syntaxes, including **INTERSECT**, **EXCEPT**, and **ROLLUP**. The TPC-DS queries range in complexity from simple to complex. Db2 Big SQL can run all 99 queries as is or with allowed minor query modifications.

Most other SQL-on-Hadoop implementations cannot do that task. Certain queries had to be rewritten because support for certain standard SQL expressions was missing. This example is of one query that had to be rewritten for a different SQL-on-Hadoop implementation. When looking at various vendors' performance numbers, it is important to understand the effort that is required to achieve those numbers, including the effort to rewrite standard SQL queries so that they could be run against the target platform.

### References:

SQL processing features (Complex queries):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_processing.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_processing.html)

## SQL capability highlights (1 of 4)

- Query operations:
  - UNION, INTERSECT, and EXCEPT
  - Wide range of built-in functions (such as aggregate functions, scalar functions, table functions, or OLAP)
- Supports three forms of a query, which are:
  - Subselect: (such as SELECT, FROM, WHERE, ORDER BY, or HAVING clauses)
  - Fullselect
  - Select-statement: (such as FOR UPDATE, OPTIMIZE FOR, or FOR READ ONLY clauses)
- Supports many predicates (such as DISTINCT, EXISTS, LIKE, or IN)

Figure 3-27. SQL capability highlights (1 of 4)

The basic relational operators that are associated with SQL are supported by Db2 Big SQL, as shown here on this and the following slide. Many more sophisticated query operations are supported as well, like SQL PL and OLAP.

**References:**

- Fullselect (**UNION**, **INTERSECT**, and **EXCEPT**):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000877.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000877.html)
- Built-in functions:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011043.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011043.html)
- OLAP specification:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0023461.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0023461.html)
- SQL queries:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008467.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008467.html)
- **CORRELATION** aggregate function:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002319.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002319.html)
- SQL processing features (compatibility with multiple SQL flavors):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview\\_processing.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview_processing.html)
- Predicates:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008495.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008495.html)

## SQL capability highlights (2 of 4)

- Supports several types of joins
  - Join operators: CROSS, INNER, LEFT OUTER, RIGHT OUTER, or FULL OUTER.
  - The order in which multiple joins are performed can affect the result.
  - Joins can be nested within other joins.
- Stored procedures and UDFs. Db2 Big SQL allows using SQL control statements, which is also called SQL Procedural Language (SQL PL)
- Cursors and flow of control (if/then/else, error handling, and others).

Figure 3-28. SQL capability highlights (2 of 4)

SQL compatibility and the ANSI SQL-compliant engine in Db2 Big SQL enables seamless transfer of applications and SQL skills to run SQL statements and procedural language (SQL PL, NZPLSQL, and PL/SQL). Db2 Big SQL is tightly integrated with the Hortonworks Data Platform (HDP) to provide a robust, reliable, and resilient environment to maximize existing business and identify new business opportunities. Db2 Big SQL is a synergetic SQL engine that offers SQL compatibility, portability, and collaborative ability to get composite analysis on data.

**References:**

- SQL processing features (compatibility with multiple SQL flavors):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview\\_processing.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/overview_processing.html)
- joined-table:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059207.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059207.html)
- Federated stored procedures:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyointfsp.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyointfsp.html)
- Db2 general UDFs:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.db2.luw.apdv.routines.commsql.doc/doc/c0003369.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.db2.luw.apdv.routines.commsql.doc/doc/c0003369.html)
- About SQL control statements (SQL Procedural Language (SQL PL)):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0008419.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0008419.html)
- Cursor values:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0054572.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0054572.html)
- IF statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0005649.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0005649.html)

## SQL capability highlights example (3 of 4)

```

SELECT
    s_name,
    count(*) AS numwait
FROM
    supplier,
    lineitem l1,
    orders,
    nation
WHERE
    s_suppkey = l1.l_suppkey
    AND o_orderkey = l1.l_orderkey
    AND o_orderstatus = 'F'
    AND l1.receiptdate > l1.commitdate
    AND EXISTS (
        SELECT
            *
        FROM
            lineitem l2
        WHERE
            l2.l_orderkey = l1.l_orderkey
            AND l2.l_suppkey <> l1.l_suppkey
    )
    AND NOT EXISTS (
        SELECT
            *
        FROM
            lineitem l3
        WHERE
            l3.l_orderkey =
l1.l_orderkey
            AND l3.l_suppkey <>
l1.l_suppkey
            AND l3.l_receiptdate >
l3.l_commitdate
    )
    AND s_nationkey = n_nationkey
    AND n_name = ':1'
GROUP BY s_name
ORDER BY numwait desc, s_name;

```

File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

Figure 3-29. SQL capability highlights example (3 of 4)

As you can see, this query is much like any standard SQL query. If you have experience with SQL, you can easily adopt Db2 Big SQL for your Hadoop data.

Look at the query: It contains various SQL expressions that you expect to find in any commercial RDBMS. But, some SQL-on-Hadoop implementations cannot run this query because it includes SQL expressions that they do not support, such as non-equi joins, subqueries in the WHERE clause, and others.

Db2 Big SQL is not case-sensitive. In other words, the following statements are all equivalent:

```

SELECT col1, col2 FROM t1;
select col1, col2 from t1;
SELECT COL1, COL2 FROM T1;

```

## SQL capability highlights (4 of 4)

Supports language features like:

- The `offset-clause` specifies the number of rows to skip before any rows are retrieved.
- The `fetch-clause` sets a maximum number of rows that can be retrieved.
- `ORDER BY` with `ASC NULLS FIRST` and `DESC NULLS FIRST`.
- User-defined aggregate functions.
- More scalar functions (such as `THIS_WEEK`, `NEXT_YEAR`, and `HASH`).

Figure 3-30. SQL capability highlights (4 of 4)

These language features provide users with richer capabilities and incorporate various popular language features. Check the references that are mentioned below for full details about the supported SQL syntax.

**References:**

- SQL queries:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008467.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0008467.html)
- offset-clause:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0061832.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0061832.html)
- fetch-clause:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059212.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059212.html)
- order-by-clause:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059211.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0059211.html)
- Migrating your SQL code to use the user-defined aggregate function:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq\\_udahowto.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/bigsq_udahowto.html)
- Scalar functions:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0000767.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0000767.html)

## A word about SQL compatibility

- **Compatibility features for Oracle:**

- Queries can use the outer join operator (+) as alternative syntax within predicates of the **WHERE** clause.
- A hierarchical query from relational data by using a **CONNECT BY** clause.
- **ROWNUM** pseudocolumn support.
- **DUAL** table support.
- **SQL\_COMPAT** global variable is used to activate the following custom Oracle compatibility features, where **SQL\_COMPAT='ORA'**:
  - **TRANSLATE** scalar function syntax
  - SQL data-access-level enforcement
  - Support for compiling and running PL/SQL statements and other language elements
  - Oracle Database link syntax
  - Synonym usage

- **Compatibility features for Netezza Platform Software (NPS):**

Several NPS compatibility features are always active, like the following ones:

- Includes a **CREATE TEMPORARY TABLE** statement.
- Queries use the outer join operator (+) as alternative syntax within predicates of a **WHERE** clause.
- Implicit casting.
- **TIMESTAMP\_FORMAT** and **VARCHAR\_FORMAT** scalar functions.
- **SQL\_COMPAT** global variable to activate the following custom NPS compatibility features, where **SQL\_COMPAT='NPS'**:
  - Double-dot notation for database objects
  - **TRANSLATE** scalar function syntax
  - NZPLSQL language in addition to the SQL PL language

*Figure 3-31. A word about SQL compatibility*

Db2 Big SQL supports certain SQL syntax and functions that are specific to popular relational DBMS offerings.

These items are designed to provide greater compatibility with Oracle and Netezza Platform Software, which make it easier for Oracle and NPS programmers to leverage their SQL skills here. This situation is important for firms seeking to offload "cold" warehouse data to Hadoop or to augment their existing warehouses with Hadoop-based data.

The built-in **SQL\_COMPAT** global variable specifies the SQL compatibility mode. Its value determines which set of syntax rules are applied to SQL queries. This global variable must have a value of **NULL**, '**DB2**', '**NPS**', or '**ORA**' (SQLSTATE 42815).

For more information about the supported SQL syntax, see the following references.

**References:**

- Compatibility features for Oracle:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/c\\_compat\\_oracle.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/c_compat_oracle.html)

- Compatibility features for Netezza Platform Software (NPS):

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.doc/doc/c\\_compat\\_netezza.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.doc/doc/c_compat_netezza.html)

- SQL\_COMPAT global variable:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/globvar\\_sql\\_compat.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/globvar_sql_compat.html)

## Functions

- Built-in functions (provided with the database manager):
  - Aggregate functions (such as AVG).
  - Operator functions (such as +).
  - Casting functions (such as DECIMAL).
  - Scalar functions (such as CEILING).
  - Table functions (such as BASE\_TABLE).
- User-defined functions:
  - Created by using an SQL data definition statement.
  - Registered to the database manager in the catalog.
  - Extend the capabilities of the database system by adding function definitions (provided by users or third-party vendors) that can be applied in the database engine itself.
  - User-defined schema functions are created by using the `CREATE FUNCTION` statement.

Figure 3-32. Functions

A function is an operation that is denoted by a function name followed by one or more operands that are enclosed in parentheses.

A function represents a relationship between a set of input values and a set of result values. The input values to a function are called *arguments*. For example, the `TIMESTAMP` function can be passed arguments of type DATE and TIME, and the result is a `TIMESTAMP`.

There are several ways to classify functions. One way is to classify functions as either built-in or user-defined:

- *Built-in functions* are functions that are provided with the database manager. Built-in functions include aggregate functions (for example, AVG), operator functions (for example, +), casting functions (for example, DECIMAL), scalar functions (for example, CEILING), and table functions (for example, BASE\_TABLE). Built-in functions are generally defined in schemas that begin with 'SYS' (for example, SYSIBM, SYSFUN, and SYSIBMADM) although some are also defined in schemas that begin with 'DB2' (for example, DB2MQ).
- *User-defined functions* are functions that are created by using an SQL data definition statement and registered to the database manager in the catalog. User-defined schema functions are created by using the **CREATE FUNCTION** statement. User-defined module functions are created by using the **ALTER MODULE ADD FUNCTION** or **ALTER MODULE PUBLISH FUNCTION** statements. A set of user-defined module functions is provided with the database manager in a set of modules in a schema called SYSIBMADM. A user-defined function is in the schema in which it was created or in the module where it was added or published. User-defined functions extend the capabilities of the database system by adding function definitions (provided by users or third-party vendors) that can be applied in the database engine itself. Extending database functions lets the database use the same functions in the engine that an application uses, providing more synergy between application and database.

### References:

- Functions:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000735.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000735.html)
- Built-in functions:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011043.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0011043.html)
- User-defined functions:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0000873.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/c0000873.html)

## Unit summary

- Described the file formats that are supported by Db2 Big SQL.
- Queried Db2 Big SQL tables by using various Data Manipulation Languages (DMLs).

## Review questions

1. Which file format is supported by Db2 Big SQL?
  - A. Text
  - B. Sequence
  - C. Parquet
  - D. ORC
  - E. Avro
  - F. All the above
  
2. Which file format provides excellent performance when used with Db2 Big SQL?
  - A. Parquet
  - B. Avro
  - C. ORC
  - D. A and B
  - E. A and C



File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

Figure 3-34. Review questions

Write your answers here:

1. F. All the above.
2. E. A and C.

## Review questions (cont.)

3. True or False: Parquet relies on schemas for communicating the structure of the data.
4. Which file format maintains extra metadata to recognize record boundaries?
  - A. Text
  - B. Sequence
  - C. Parquet
  - D. ORC
  - E. Avro
5. True or False: The ORC format uses type-specific encoders for each column and divides the file into large stripes.



Figure 3-35. Review questions (cont.)

Write your answers here:

3. False. Avro relies on schemas for communicating the structure of the data.
4. B.
5. True.

## Review answers

1. Which file format is supported by Db2 Big SQL?
  - A. Text
  - B. Sequence
  - C. Parquet
  - D. ORC
  - E. Avro
  - F. **All the above**
  
2. Which file format provides excellent performance when used with Db2 Big SQL?
  - A. Parquet
  - B. Avro
  - C. ORC
  - D. A and B
  - E. **A and C**



## Review answers (cont.)

3. True or False: Parquet relies on schemas for communicating the structure of the data.
4. Which file format maintains extra metadata to recognize record boundaries?
  - A. Text
  - B. Sequence
  - C. Parquet
  - D. ORC
  - E. Avro.
5. True or False: The ORC format uses type-specific encoders for each column and divides the file into large stripes.



## Exercise: Querying Db2 Big SQL tables

File formats and querying IBM Db2 Big SQL tables

© Copyright IBM Corporation 2021

*Figure 3-38. Exercise: Querying Db2 Big SQL tables*

## Exercise objectives

- In this exercise, you experiment with advanced SQL queries. Then, you explore the Db2 Big SQL ARRAY type. You also create a user-defined function (UDF) and write queries that call the UDF. Finally, you store data in an alternative file format (Parquet).
- After completing this exercise, you should be able to:
  - Run more advanced Db2 Big SQL queries.
  - Work with ARRAY data type.
  - Create user-defined functions (UDFs).
  - Store data in popular formats for Hadoop environments such as Parquet.



Figure 3-39. Exercise objectives

---

# Unit 4. Configuring IBM Db2 Big SQL security

## Estimated time

01:00

## Overview

In this unit, you learn how to configure Db2 Big SQL security.

## Unit objectives

- Configure authentication for Big SQL.
- Manage security by using Apache Ranger.
- Enable SSL encryption.
- Configure authorization of Big SQL objects.
- Configure impersonation in Big SQL.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-1. Unit objectives*

## 4.1. Db2 Big SQL encryption features

## Db2 Big SQL encryption features

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-2. Db2 Big SQL encryption features*

## Topics

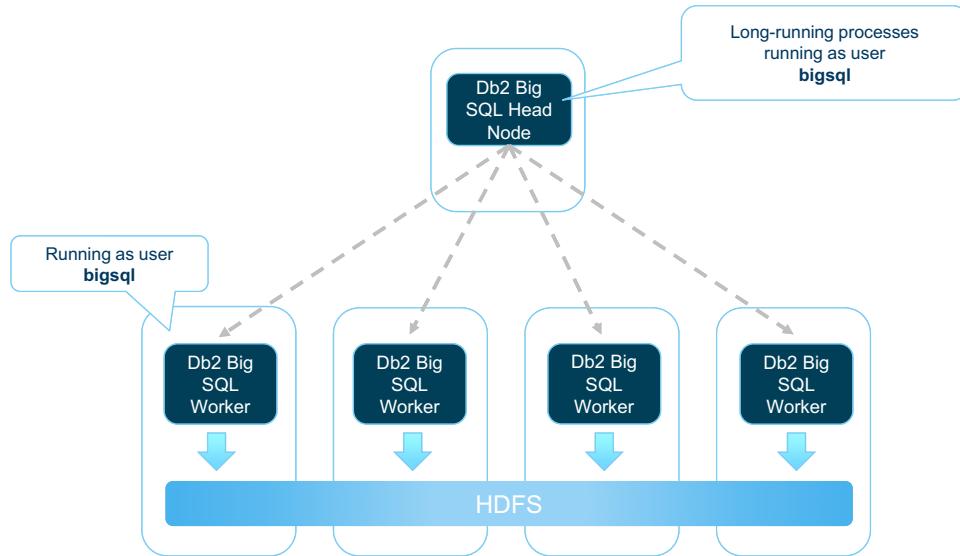
-  Db2 Big SQL encryption features
  - Db2 Big SQL authentication
  - Apache Ranger security for Db2 Big SQL
  - Native Db2 Big SQL authorization
  - Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-3. Topics*

## Basic process architecture



[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-4. Basic process architecture

Db2 Big SQL can operate under different security models depending on the use case. To understand these security models, here is a high-level architecture overview. Db2 Big SQL is classified as a long-running service in Hadoop, which means that the service is running whether users are connected to Db2 Big SQL. This operating model is what provides Db2 Big SQL the ability to support low latency and high concurrency workloads.

The Db2 Big SQL head node has a set of processes running and waiting to perform work. These operating system processes are running as a service ID called `bigsq1`, by default.

Similarly, on the data nodes, Db2 Big SQL worker processes are running and waiting to perform work. These processes are also running as the `bigsq1` service ID. The Db2 Big SQL workers perform the reads and writes as the `bigsq1` user to HDFS when queries are submitted.

Db2 Big SQL can operate under different security models depending on the use case. Security in Db2 Big SQL can be based on database authorizations (authorizations that are enforced by the database engine) or based on Hadoop security (authorizations that are enforced by the Apache Hadoop stack, such as HDFS privileges and permissions, and Apache Ranger policies).

**Reference:**

Security and governance:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bigsql\\_securityconsider.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bigsql_securityconsider.html)

## Encryption features

- IBM Db2 Big SQL includes encryption features for:
  - Data in-transit (SSL)
  - HDFS data at-rest encryption (transparent data encryption (TDE) and Apache Ranger KMS).
- You can encrypt selected files or directories in HDFS, which saves on overhead and protects performance.
- Any database client application that also supports SSL can connect to the Db2 Big SQL server over an SSL socket.
- If the Db2 Big SQL server is running on both a non-SSL and an SSL ports:
  - The priority goes to the SSL connection if the end connection is coming from https.
  - The priority goes to a non-SSL connection if the end connection is coming from http.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-5. Encryption features

IBM Db2 Big SQL includes encryption features for data in-transit (SSL) and uses HDFS data at-rest encryption (transparent data encryption (TDE) and Apache Ranger KMS).

You can encrypt selected files or directories in HDFS, which saves on overhead and protects performance. HDFS TDE takes advantage of HDFS native data encryption without any application code changes.

Because Db2 Big SQL server supports SSL, any database client application that also supports SSL can connect to the Db2 Big SQL server over an SSL socket. Applications that use the IBM Data Server Driver for JDBC and SQLJ (type 4 connections) support SSL.

To enable SSL encryption (data in motion), complete the following steps:

1. Configure and enable SSL on the server by following the steps in IBM Knowledge Center:  
[https://www.ibm.com/support/knowledgecenter/SSEPGG\\_11.5.0/com.ibm.db2.luw.admin.sec.doc/doc/t0025241.html](https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.admin.sec.doc/doc/t0025241.html)
2. Copy the following files to the worker nodes from the head node:  
`<home_directory>/sqllib/security/keystore/mydbserver.kdb`  
`<home_directory>/sqllib/security/keystore/mydbserver.sth`  
The `home_directory` refers to the home directory of the Db2 Big SQL service user.
3. Enable SSL in the client.

If the Db2 Big SQL server is running on both a non-SSL and an SSL port, then the priority goes to the SSL connection if the end connection is coming from https. The priority goes to a non-SSL connection if the end connection is coming from http.

### References:

- Encryption:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/admin\\_nav\\_encryption.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/admin_nav_encryption.html)
- Enabling SSL (Secure Socket Layer) encryption (data in motion):  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bi\\_admin\\_bigA\\_ssl.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bi_admin_bigA_ssl.html)

## 4.2. Db2 Big SQL authentication

## Db2 Big SQL authentication

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-6. Db2 Big SQL authentication*

## Topics

- Db2 Big SQL encryption features
- Db2 Big SQL authentication
  - Apache Ranger security for Db2 Big SQL
  - Native Db2 Big SQL authorization
  - Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-7. Topics

## Authentication for Db2 Big SQL

- Pluggable Authentication Module (PAM) authentication:
  - Db2 Big SQL can support authentication by using PAM.
  - Db2 Big SQL can use LDAP to authenticate users.
  - PAM can be used in addition to client authentication, which can be done through LDAP.
  - It is also possible to use a different PAM file than the one that Db2 Big SQL uses by default.
- Kerberos authentication:
  - Kerberos is a third-party authentication mechanism.
  - Users and services rely on the Kerberos server or the Key Distribution Center (KDC) to authenticate to each other.
  - Central repository for IDs (also known as principals).
  - Db2 Big SQL introduces its own service principal and uses keytab files to store credentials for authentication.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

*Figure 4-8. Authentication for Db2 Big SQL*

Authentication is the process by which a system verifies the identity of a user. The authentication process produces a Db2 Big SQL authorization ID user in the Hortonworks Data Platform (HDP) administrative group. Groups are a convenient way to authorize a collection of users.

You can configure authentication security for Db2 Big SQL by using either of the following methods:

- Enabling PAM authentication for Db2 Big SQL:

The authentication of users and group lookup is done through the operating system, so Db2 Big SQL can support authentication by using the Pluggable Authentication Module (PAM) method.

Regarding authentication modes, Db2 Big SQL can use LDAP to authenticate users. This authentication is in addition to client authentication, which can also be done through LDAP. It is also possible to use a different PAM file than the one that Db2 Big SQL uses by default.

Normally, the Db2 Big SQL installer uses the appropriate PAM file that is supplied in the installation image.

- Enabling Kerberos authentication:

Hadoop uses Kerberos as the basis for strong authentication and identity propagation for both users and services. Kerberos is a third-party authentication mechanism in which users and services rely on a third party, such as the Kerberos server or the Key Distribution Center (KDC) to authenticate to each other. By using Kerberos for authentication, you use a central repository for IDs, also known as principals. Db2 Big SQL introduces its own service principal and uses keytab files to store credentials for authentication. Enabling Kerberos for the cluster and its services, including Db2 Big SQL, is necessary for securing the cluster.

Enabling Kerberos for Db2 Big SQL clients is separate and can be optionally set up by following the steps at IBM Knowledge Center:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...#admin\\_kerb\\_configure](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## References:

- Security and governance:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
bigsql\\_securityconsider.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Enabling PAM authentication for Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
bi\\_admin\\_big\\_a\\_enable\\_authentication.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Enabling Kerberos authentication:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
admin\\_nav\\_kerberos.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

- Setting up Kerberos for the Db2 Big SQL service:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
admin\\_bigsq...kerberos.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Setting up Kerberos for the Db2 Big SQL service

- Enable Kerberos security in the Ambari admin dashboard.
- Configure Kerberos to work in the Db2 Big SQL service:
  - “Kerberize” the cluster after installing Db2 Big SQL.
  - Install Db2 Big SQL on a Kerberized cluster.
- Kerberos troubleshooting commands:
  - Start Kerberos by running the following commands:  
 /sbin/service krb5kdc start  
 /sbin/service kadmin start
  - Run kadmin commands by using kadmin.local.
  - Get detailed information about a principal from the Kerberos database by running the following command on kadmin:  
 getprinc bigsql/myserver.abc.com@MYCOMPANY.COM
  - Set the max\_renewable\_life parameter in the realms section:  
 /var/kerberos/krb5kdc/kdc.conf

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-9. Setting up Kerberos for the Db2 Big SQL service

You can set up Kerberos for the Db2 Big SQL service. First, prepare the cluster by following the instructions that are provided by HDP documentation to enable Kerberos security in the Ambari admin dashboard:

[https://docs.hortonworks.com/HDPDocuments/Ambari-2.5.1.0/bk\\_ambari-security/content/enabling\\_kerberos\\_security\\_in\\_ambari.html](https://docs.hortonworks.com/HDPDocuments/Ambari-2.5.1.0/bk_ambari-security/content/enabling_kerberos_security_in_ambari.html)

The default settings should be sufficient, but you can review and modify them for your specific requirements.

You can install Db2 Big SQL on a Kerberized cluster or Kerberize the cluster after installing Db2 Big SQL. For either case, you can choose the automated Kerberos setup for Db2 Big SQL by using the Ambari wizard or you can choose to manage Kerberos principals and keytabs manually.

Check the following Kerberos troubleshooting commands:

- Start Kerberos by running the following commands. For more information, see [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/6/html/managing\\_smart\\_cards/configuring\\_a\\_kerberos\\_5\\_server](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/managing_smart_cards/configuring_a_kerberos_5_server).
 

```
/sbin/service krb5kdc start  
/sbin/service kadmin start
```
- kadmin and kadmin.local are command-line interfaces that are used for Kerberos V5 administration system. kadmin.local is used for direct access to the KDC database, and kadmin performs operations by using kadmind. For more information, see the following website: [https://web.mit.edu/kerberos/krb5-1.12/doc/admin/admin\\_commands/kadmin\\_local.html](https://web.mit.edu/kerberos/krb5-1.12/doc/admin/admin_commands/kadmin_local.html)
- The **getprinc** command gets detailed information about the existing principal in the Kerberos database. The input parameter for the **getprinc** command is the principal name that matches the one that exists in the Kerberos database. For more information, see the following website: <https://www.oreilly.com/library/view/kerberos-the-definitive/0596004036/re02.html>
- When running Db2 Big SQL statements on a Kerberized cluster, you might encounter Kerberos KDC errors about credentials renewal in the bigsql.log file because error occurs when the KDC fails to generate a renewable ticket-granting ticket (TGT). Ensure that a TGT is generated and renewed while Db2 Big SQL processes are running by setting the *max\_renewable\_life* parameter in the realms section of the /var/kerberos/krb5kdc/kdc.conf file on the Kerberos server. This step is not required on the client side.

### References:

- Setting up Kerberos for the Db2 Big SQL service:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin\\_bigsqld\\_kerberos.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin_bigsqld_kerberos.html)
- "KDC can't fulfill requested option while renewing credentials" errors when running Db2 Big SQL statements on a Kerberized cluster:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/trb\\_kerb\\_kdc.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/trb_kerb_kdc.html)

## 4.3. Apache Ranger security for Db2 Big SQL

## Apache Ranger security for Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-10. Apache Ranger security for Db2 Big SQL*

## Topics

- Db2 Big SQL encryption features
- Db2 Big SQL authentication
- Apache Ranger security for Db2 Big SQL
  - Native Db2 Big SQL authorization
  - Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-11. Topics

## Authorization that uses the Db2 Big SQL Apache Ranger plug-in

- Apache Ranger is a framework to enable, monitor, and manage comprehensive data security across the Hadoop platform.
- Using the Db2 Big SQL Apache Ranger plug-in, a security administrator can create policies to grant access to tables and views in a Db2 Big SQL database.
- These Apache Ranger policies grant and restrict access in a manner that is like native Db2 Big SQL table and view privileges.
- Enable or disable the Db2 Big SQL Apache Ranger plug-in.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-12. Authorization that uses the Db2 Big SQL Apache Ranger plug-in

Security in Db2 Big SQL can be based on database authorizations (authorizations that are enforced by the database engine) or based on Hadoop security (authorizations that are enforced by the Apache Hadoop stack, such as HDFS privileges and permissions, and Apache Ranger policies).

There are two available types for authorization security at Db2 Big SQL:

- Native Db2 Big SQL authorization
- Authorization that uses the Db2 Big SQL Apache Ranger plug-in.

Apache Ranger is a framework to enable, monitor, and manage comprehensive data security across the Hadoop platform.

Apache Ranger can be installed either manually by using the HDP or the Ambari User Interface (UI).

You must first install Apache Ranger, and then you can enable the Db2 Big SQL Apache Ranger plug-in. After the plug-in is enabled, Apache Ranger can audit all access to Db2 Big SQL tables, views, and nicknames. Db2 Big SQL native authorization controls can be used for other database objects.

Disabling the Db2 Big SQL Apache Ranger plug-in deletes all existing policies. Before disabling the plug-in, you might want to export the policies by using the export feature in the Apache Ranger UI. Also, disabling the plug-in leads causes the system to revert to using native Db2 Big SQL authorization controls. Native Db2 Big SQL security authorizations should be examined in detail to ensure that only necessary object access is granted.

One key difference is when the Db2 Big SQL Apache Ranger plug-in is unavailable, then users have full access to objects they own. This situation is different from when the Db2 Big SQL Apache Ranger plug-in was enabled, where an Apache Ranger policy was required to grant a user access to an object that they own.

### References:

- Security and governance:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bigsql\\_securityconsider.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bigsql_securityconsider.html)

- Authorization:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_nav\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_nav_authorization.html)

- Apache Ranger:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_nav\\_ranger.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_nav_ranger.html)

- Enabling and disabling Apache Ranger security support for Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bigsql\\_ranger\\_enable.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bigsql_ranger_enable.html)

- Authorization:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_nav\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_nav_authorization.html)

## Database operations that the Apache Ranger plug-in controls

Operation on Db2 Big SQL tables	Permission that is required in the Apache Ranger policy
Select from table	Select on schema or table.
Select from view	Select on schema or view.
Insert	Insert on schema or table.
Delete	Delete on schema or table.
Update	Update on schema or table.
Truncate	Delete on schema or table.
Alter	Alter on schema or table.
Rename	Alter on schema or table.
Create Table	Create on schema or table. Create on schema required for HBase tables.
Create Index	Create on schema. Index on schema or table.
Drop	Drop on schema or table.
Analyze	Analyze on schema or table.
Msck Repair	Alter on schema or table.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-13. Database operations that the Apache Ranger plug-in controls

After Apache Ranger is enabled for Db2 Big SQL, there are two authorization control systems: Apache Ranger and native Db2 Big SQL. For operations that are managed by Apache Ranger, the Db2 Big SQL native authorization controls are no longer used. For operations that are not managed by Apache Ranger, the Big SQL native authorization controls are used.

For the Analyze command, in addition to Analyze on table in Apache Ranger, a user also needs whatever permission it needs in native Db2 Big SQL.

### Reference:

Operations managed by the Db2 Big SQL Apache Ranger plug-in:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../big\\_sq...\\_ranger\\_operations.html#bigsq...\\_ranger\\_operations](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../big_sq..._ranger_operations.html#bigsq..._ranger_operations)

## 4.4. Native Db2 Big SQL authorization

## Native Db2 Big SQL authorization

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-14. Native Db2 Big SQL authorization*

## Topics

- Db2 Big SQL encryption features
- Db2 Big SQL authentication
- Apache Ranger security for Db2 Big SQL
- Native Db2 Big SQL authorization
- Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-15. Topics

## Authorization of Db2 Big SQL objects

- Level 1: Controlling access with authorization in the distributed file system
- Level 2: Authorization with the GRANT command
- Level 3: Authorization at the row and column level
- Level 4: Controlling access by using VIEWS or STORED PROCEDURES.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-16. Authorization of Db2 Big SQL objects

Authorization of Db2 Big SQL objects is controlled at several layers. The Db2 Big SQL permissions are maintained within the Db2 Big SQL catalog tables.

### Reference:

Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
admin\\_big...\\_enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Level 1: Controlling access with authorization in the distributed file system

- Db2 Big SQL server runs as the bigsql user and accesses tables and data as the bigsql user.
- Creating a schema:
  - A directory is created inside /apps/hive/warehouse in the distributed file system (DFS).
  - The schema directory is created with the permissions that are inherited from the /apps/hive/warehouse directory.
- Directory permissions: Set the directory permissions to the DFS first.
  - Set the permissions to 770 to give full access to the owner and the group but provide no public access by running the following command:  
`hdfs dfs -chmod 770 /apps/hive/warehouse`
  - Set the permissions to 700 to give full access for the bigsql user but no access for anyone else by running the following command:  
`hdfs dfs -chmod 700 /apps/hive/warehouse`

*Figure 4-17. Level 1: Controlling access with authorization in the distributed file system*

Level 1 is controlling access with authorization in the distributed file system (DFS), where the Db2 Big SQL server runs as the bigsql user and accesses tables and data as the bigsql user. In this context, the bigsql user is the database administrator.

By default, the bigsql user is in the admin group only. Because the sysadm\_group authority is assigned to the primary user group of bigsql, you might also want to create the bigsql user in a primary group to which only bigsql belongs.

At the file system level, you must have Linux-style privileges to access the directories and files that contain tables and the data that is contained in those tables.

For example, when you create a schema in the Db2 Big SQL server, Db2 Big SQL creates a directory inside the /apps/hive/warehouse directory in the DFS. That schema directory is created with the permissions that are inherited from the /apps/hive/warehouse directory. The /apps/hive/warehouse directory is the standard container for all schema and table objects in Hadoop.

So, before you create any schemas, you should set the directory permissions to the DFS first. This way, when you create the schemas, they automatically inherit what you set.

The slide lists two examples:

- Set the permissions to 770 to give full access by the owner and the group but provide no public access by running the following command:

```
hdfs dfs -chmod 770 /apps/hive/warehouse
```

- Set the permissions to 700 to give full access for the bigsql user, but no access for anyone else by running the following command:

```
hdfs dfs -chmod 700 /apps/hive/warehouse
```

This permission level is appropriate if you want the bigsql user to guarantee access control. In this scenario, the bigsql user is your only access to the data, so the bigsql user issues the **GRANT** and **REVOKE** permissions, as needed.

**Reference:**

Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bi\\_admin\\_biga\\_enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/bi_admin_biga_enable_authorization.html)

## Level 2: Authorization with the GRANT command

- Use **GRANT** or **REVOKE** commands to give or remove certain privileges to users.
- If you created the table, you automatically have the proper authority.
- The DBA can grant authority to other users. For example, an administrator is granting a user that is named user1 access to a table that is called SALES:

```
GRANT INSERT, SELECT, DELETE ON myschema.SALES TO USER user1
```

*Figure 4-18. Level 2: Authorization with the GRANT command*

A **privilege** is a permission to perform an action or a task. If you have the correct authorization, you can create objects and access them. In addition, as the owner and creator of the object, you can use the **GRANT** command to give a privilege to other users.

The security of **GRANT** and **REVOKE** depends on the Db2 Big SQL server security. You use **GRANT** and **REVOKE** commands to give or remove certain privileges to users. If you do not have database administrative authority, you cannot load data into a table. If you created the table, you automatically have the proper authority. If you are the database administrator, you can grant authority to other users.

In the following example, an administrator is granting a user that is named user1 access to a table that is called SALES:

```
GRANT INSERT, SELECT, DELETE ON myschema.SALES TO USER user1
```

### Reference:

Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/bi\\_admin\\_big...\\_enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/bi_admin_big..._enable_authorization.html)

## Level 3: Authorization at the row and column level

- Row and column access control (RCAC) provides an extra layer of security to privileges.
- RCAC controls access to a table at the row level, column level, or both.
- The row permission rule is in the form of an SQL search condition that describes the set of rows to which a user has access.
- The column mask rule is an SQL **CASE** expression that describes the column values that a user is permitted to see and under what conditions.
- You should be aware that queries do not see the whole picture in terms of the data in the table unless granted specific permission to do so.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-19. Level 3: Authorization at the row and column level

Row and column access control (RCAC) provides an extra layer of security to privileges. RCAC controls access to a table at the row level, column level, or both. The row permission rule is in the form of an SQL search condition that describes the set of rows to which a user has access. The column mask rule is an SQL **CASE** expression that describes the column values that a user is permitted to see and under what conditions.

You do not need to change your application to take advantage of RCAC. RCAC is an important security advantage in using Db2 Big SQL because the security administrator controls the access, not the SQL application, but you should be aware that queries do not see the whole picture in terms of the data in the table unless granted specific permission to do so.

### Reference:

Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../admin\\_big.../enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../admin_big.../enable_authorization.html)

## Level 3: Why use row and column access control

- Managing privileges for users is tedious and error-prone:
  - New users must decide what kind of access they need.
  - Departing users must clean up their access rights.
- Users should see only data that matters to them (and not see data that they should not).
- Sensitive data in columns.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-20. Level 3: Why use row and column access control

Before you get into the details of how to implement RCAC, look at some of the reasons why you should use RCAC:

- Managing privileges on users is tedious and error prone. Imagine that you must grant six different privileges across five tables for two users for a total of 12 **GRANT** statements that you must manage for only two users across five tables. Now, the problem magnifies if you had to manage it for hundreds of users across dozens of tables. New users must decide what kind of access they need. Departing users must clean up their access rights.
- Users should see only the data that matters to them (and not see the data that they should not). Say that there are three departments or branches, and if you belong to a particular branch, you should see only the data pertaining to that branch and not the other two.
- Sensitive data in columns. A good and common example is salary information. Most users should not have access to salary information, so it must be hidden from other users.

## Level 3: Advantages of row and column access control

- No database user is inherently exempted from the RCAC rules.
- Table data is protected regardless of how a table is accessed through SQL.
- No application changes are required to take advantage of this additional layer of data security.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-21. Level 3: Advantages of row and column access control

Before you get into the details about how to implement RCAC, review the following three advantages of using RCAC:

- No database user is inherently exempted from the RCAC rules.

Even higher-level authorities such as users with DATAACCESS authority are not exempt from these rules. Only users with security administrator (SECADM) authority can manage RCACs within a database. Therefore, you can use RCAC to prevent users with DATAACCESS authority from freely accessing all data in a database.

- Table data is protected regardless of how a table is accessed through SQL.

Applications, improvised query tools, and report generation tools are all subject to RCAC rules. The enforcement is data-centric.

- No application changes are required to take advantage of this additional layer of data security.

Row and column level access controls are established and defined in a way that is not apparent to existing applications. However, RCAC represents an important shift in paradigm in the sense that it is no longer what is being asked but rather who is asking what. Result sets for the same query change based on the context in which the query was asked, and there is no warning or error returned. This behavior is the exact intent of the solution. It means that application designers, DBAs, and data analysts must be conscious that queries do not see the whole picture in terms of the data in the table, unless granted specific permissions to do so.

So, you can use RCAC to ensure that your users have access to only the data that is required for their work. For example, a hospital running Db2 together with Db2 Big SQL might use RCAC to filter patient information and data to include only that data that a particular doctor requires. Other patients do not exist as far as the doctor is concerned. Similarly, when a patient service representative queries the patient table at the same hospital, they can view the patient name and telephone number columns, but the medical history column is masked for them. If data is masked, a NULL or an alternative value is displayed instead of the actual medical history.

Also, RCAC can be used with the Db2 Big SQL Apache Ranger plug-in to control authorizations on Db2 Big SQL objects, which provide an optimal security solution.

### **References:**

RCAC overview:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin\\_rcac\\_overview.html#admin\\_rcac\\_overview](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/admin_rcac_overview.html#admin_rcac_overview)

## Level 3: Row-based access control (1 of 5)

Sample data to show row-based access control.

```
SELECT * FROM BRANCH_TBL

EMP_NO    FIRST_NAME      BRANCH_NAME
-----  -----
1  Steve          Branch_B
2  Chris          Branch_A
3  Paula          Branch_A
4  Craig          Branch_B
5  Pete           Branch_A
6  Stephanie      Branch_B
7  Julie          Branch_B
8  Chrissie       Branch_A
8 record(s) selected.
```

Figure 4-22. Level 3: Row-based access control (1 of 5)

Here is some sample data to show row-based access control. Select \* from the BRANCH\_TBL table. Eight rows are returned. Four of those rows belong to Branch\_A and the other four belong to Branch\_B. What do you need to do to limit the rows to either Branch\_A or Branch\_B?

## Level 3: Row-based access control (2 of 5)

**CREATE** and **GRANT** access and roles:

- **CREATE** a BRANCH\_A\_ROLE.
- **GRANT** that role to USER newton.
- **GRANT SELECT** to that role.

```
CREATE ROLE BRANCH_A_ROLE
GRANT ROLE BRANCH_A_ROLE TO USER newton
GRANT SELECT ON BRANCH_TBL TO USER newton
                           ROLE BRANCH_A_ROLE
```

Figure 4-23. Level 3: Row-based access control (2 of 5)

Say that you have a user that is called “newton” that must see only results from Branch\_A. Issue **CREATE ROLE** BRANCH\_A\_ROLE, and then **GRANT** this role to the USER “newton”. Finally, instead of granting the **SELECT** privilege to the user, you grant it to the role.

Now, imagine that if you had multiple privileges that you had to set, you would need to set them once for the role, and then as new users need them, you assign the new users to that role. This setup greatly simplifies the case where if you had hundreds of users, you need to assign only the role to them (as opposed to each individual privilege). The reverse is also true where if a user must have their access revoked, you remove them from only the role (as opposed to revoking each privilege).

**References:**

- **CREATE ROLE** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0050615.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0050615.html)

- **GRANT** (table, view, or nickname privileges) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000966.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0000966.html)

- **GRANT** (role) statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0050616.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0050616.html)

## Level 3: Row-based access control (3 of 5)

Create permissions: `CREATE PERMISSION BRANCH_A_ACCESS` on the table:

- The user is equal to `BRANCH_A_ROLE`.
- The branch is equal to `BRANCH_A`.

```
CREATE PERMISSION BRANCH_A_ACCESS ON BRANCH_TBL
FOR ROWS WHERE(VERIFY_ROLE_FOR_USER(SESSION_USER, 'BRANCH_A_ROLE') = 1
AND
BRANCH_TBL.BRANCH_NAME = 'Branch_A')
ENFORCED FOR ALL ACCESS
ENABLE
```

*Figure 4-24. Level 3: Row-based access control (3 of 5)*

Create the permission for `BRANCH_A_ACCESS` on the `BRANCH_TBL`, where if the user is in `BRANCH_A_ROLE`, then return only the rows that are pertaining to Branch\_A.

### Reference:

#### **CREATE PERMISSION** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/r0057429.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/r0057429.html)

## Level 3: Row-based access control (4 of 5)

Enable access control: Activate the access control on the table.

```
ALTER TABLE BRANCH_TBL ACTIVATE ROW ACCESS CONTROL
```

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-25. Level 3: Row-based access control (4 of 5)

Enable the row access control on the BRANCH\_TBL table.

**Reference:**

**ALTER TABLE** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq...  
c/doc/r0000888.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq...)

## Level 3: Row-based access control (5 of 5)

**SELECT as a USER of BRANCH\_A\_ROLE:** Only the four records from Branch\_A are returned.

```
CONNECT TO TESTDB USER newton
SELECT "*" FROM BRANCH_TBL

EMP_NO      FIRST_NAME      BRANCH_NAME
-----      -----
2 Chris      Branch_A
3 Paula      Branch_A
5 Pete       Branch_A
8 Chrissie   Branch_A
4 record(s) selected.
```

Figure 4-26. Level 3: Row-based access control (5 of 5)

Perform another **SELECT** on the table as a USER of the BRANCH\_A\_ROLE, which is “newton” in the example. Only the Branch\_A rows are returned because of the row access control. Now, if the user newton must have access revoked, a simple revoke from the BRANCH\_A\_ROLE removes the user’s privilege to select from Branch\_A. This action solves the problem where managing users is tedious and error-prone, and users should see the data that applies only to them.

## Level 3: Column-based access control (1 of 5)

- Data from the salary table
- Three columns:
  - EMP\_NO
  - FIRST\_NAME
  - SALARY

```
SELECT *** FROM SAL_TBL

EMP_NO    FIRST_NAME      SALARY
-----  -----
1  Steve          250000
2  Chris          200000
3  Paula         1000000
```

Figure 4-27. Level 3: Column-based access control (1 of 5)

Here is a table of salary information to demonstrate column-based access control. There are three columns in this table: employee number, first name, and the salary. You are going to hide (mask) the salary column from employees but keep it visible to managers.

## Level 3: Column-based access control (2 of 5)

- Create and grant access and roles.
- Create a manager and an employee role and grant it to two different users.

```
CREATE ROLE MANAGER  
CREATE ROLE EMPLOYEE  
  
GRANT SELECT ON SAL_TBL TO ROLE MANAGER  
GRANT SELECT ON SAL_TBL TO ROLE EMPLOYEE  
  
GRANT ROLE MANAGER TO USER socrates  
GRANT ROLE EMPLOYEE TO USER newton
```

Figure 4-28. Level 3: Column-based access control (2 of 5)

First, create two roles: MANAGER and EMPLOYEE. Then, **GRANT** the **SELECT** privilege to the two different roles: MANAGER and EMPLOYEE. Finally, **GRANT** the MANAGER role to USER socrates and the EMPLOYEE role to USER newton. So, socrates is the MANAGER, and newton is the EMPLOYEE.

## Level 3: Column-based access control (3 of 5)

Create permissions by creating a mask on the SALARY column:

- If the user belongs to the MANAGER role, show the SALARY column.
- Otherwise, show 0.00.

```
CREATE MASK SALARY_MASK ON SAL_TBL FOR
COLUMN SALARY RETURN
CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER, 'MANAGER') = 1
THEN SALARY
ELSE 0.00
END
ENABLE
```

Figure 4-29. Level 3: Column-based access control (3 of 5)

Create the SALARY\_MASK on the SAL\_TBL on the SALARY column. If the user belongs to the MANAGER role, show the SALARY column. Otherwise, show the value 0.00.

### Reference:

#### CREATE MASK statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/r0058564.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/r0058564.html)

## Level 3: Column-based access control (4 of 5)

Enable access control by using a user with SECADM authority.

```
ALTER TABLE SAL_TBL ACTIVATE COLUMN ACCESS CONTROL
```

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-30. Level 3: Column-based access control (4 of 5)

Activate the column access control on the SAL\_TBL table.

To alter a table to ACTIVATE and DEACTIVATE RCAC, the privileges that are held by the authorization ID of the statement must include the SECADM authority.

### Reference:

**ALTER TABLE** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/r0000888.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.commsql.doc/r0000888.html)

## Level 3: Column-based access control (5 of 5)

```
CONNECT TO TESTDB USER newton
SELECT "*" FROM SAL_TBL
```

EMP_NO	FIRST_NAME	SALARY
1	Steve	0.00
2	Chris	0.00
3	Paula	0.00

3 record(s) selected.

SELECT as an EMPLOYEE.

SELECT as a MANAGER.

```
CONNECT TO TESTDB USER socrates
SELECT "*" FROM SAL_TBL
```

EMP_NO	FIRST_NAME	SALARY
1	Steve	250000
2	Chris	200000
3	Paula	1000000

3 record(s) selected.

Figure 4-31. Level 3: Column-based access control (5 of 5)

In the top example of the slide, you select from the table as the **newton** user. In the bottom example, you select as the **socrates** user. The **socrates** user has the **MANAGER** role, so the **SALARY** column is returned. The **newton** user does not have the **MANAGER** role, so the **SALARY** column is masked with the value that is provided when you set up the column masking.

## Level 4: Controlling access by using VIEWS or STORED PROCEDURES

- You can set privileges on the VIEW objects or on the STORED PROCEDURES to control access.
- With VIEWS and STORED PROCEDURE, you can allow restricted access to data to which the user normally does not have access.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-32. Level 4: Controlling access by using VIEWS or STORED PROCEDURES

You can also control access by using VIEWS or STORED PROCEDURES. You can set privileges on the VIEW objects or on the STORED PROCEDURES to control access. With VIEWS and STORED PROCEDURE, you can allow restricted access to data to which the user normally does not have access.

### Reference:

Authorization of Db2 Big SQL objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../doc/doc/bi\\_admin\\_biga\\_enable\\_authorization.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../doc/doc/bi_admin_biga_enable_authorization.html)

## 4.5. Impersonation in Db2 Big SQL

## Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-33. Impersonation in Db2 Big SQL*

## Topics

- Db2 Big SQL encryption features
  - Db2 Big SQL authentication
  - Apache Ranger security for Db2 Big SQL
  - Native Db2 Big SQL authorization
-  Impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-34. Topics

## Impersonation in Db2 Big SQL

- Impersonation is ability to allow a service user to securely access data in Hadoop on behalf of another user. For example, if impersonation is enabled, the bigsql user can impersonate the connected user to perform actions on Hadoop tables
- By default, the bigsql user performs the read/write operations on HDFS, Apache Hive, and HBase that are required for the Db2 Big SQL service. Impersonation is not needed if the bigsql user is the sole owner.
- You might need impersonation if the data that you want to analyze is produced outside of the Db2 Big SQL service:
  - Loading data from a different data source
  - Sharing data between multiple services in the cluster
- Thorough review of the entire lifecycle of your data is needed before you decide on whether to use impersonation.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

*Figure 4-35. Impersonation in Db2 Big SQL*

Impersonation is the ability to allow a service user to securely access data in Hadoop on behalf of another user. If you enable impersonation at the global level in Db2 Big SQL, the bigsql user can impersonate the connected user to perform actions on Hadoop tables. For example, if the service user must access a set of data belonging to user1, impersonation allows bigsql to access and perform actions on the data belonging to user1. Also, when you issue **CREATE HADOOP TABLE**, run a query, or load an operation, Db2 Big SQL performs the operations in HDFS and Apache Hive as the connected user.

By default, the bigsql user performs the read and write operations on HDFS, Apache Hive, and HBase that are required for the Db2 Big SQL service. You do not need impersonation if the bigsql user is the sole owner and the user of the data.

You might need impersonation if the data that you want to analyze is produced outside of the Db2 Big SQL service, or if there is sharing of data between multiple services in your cluster.

If you enable impersonation, other users must have permissions on the tables in HDFS to perform I/O operations. Therefore, there is a potential for loss of granularity in authorization control when impersonation is enabled, like the Apache Hive impersonation behavior.

A thorough review of your entire lifecycle of the data is needed before you decide on whether to use impersonation. The change to go back from impersonation is not trivial. If you turn off impersonation after running statements on tables with impersonation on, you must remember to update the HDFS and Apache Hive configurations before you access those tables again.

**Reference:**

Impersonation in Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsql\\_imperonate.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsql_imperonate.html)

## Why would you want to use impersonation

- Applies only to certain use cases.
- Data that is produced by one service is used by another one.
- Multiple services must use the same data:
  - Extract, transform, and load (ETL) moves the data into HDFS.
  - Apache Spark for some cleansing and basic analysis.
  - Db2 Big SQL for advanced analytics.
- Difficult to set up authorizations for each individual service.
- When you do not need enhanced security controls that are provided by Db2 Big SQL, like the following examples:
  - RCAC.
  - Label-based security.
  - View-based security.

[Configuring IBM Db2 Big SQL security](#)

© Copyright IBM Corporation 2021

Figure 4-36. Why would you want to use impersonation

Here are a few more reasons why you might want to use impersonation. These reasons apply only to certain use cases, so be sure to evaluate your data before deciding.

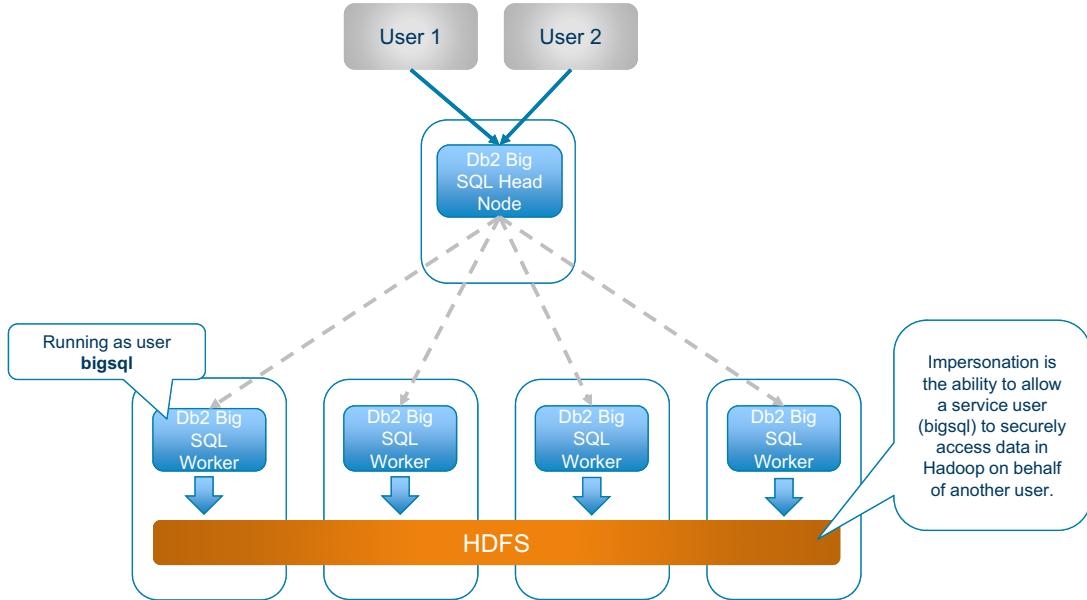
- If your data was produced by another service, particularly outside of Db2 Big SQL, you might want to use impersonation.
- If your data must be shared across multiple services, you might want to use impersonation to simplify access control.
- If it is difficult to set up authorizations for each individual service separately, you might want to use impersonation.
- When you do not need the enhanced security controls that are provided by Db2 Big SQL such as RCAC, label-based security, or view-based security. Impersonation does not work with these controls.

### Reference:

Label-based access control (LBAC) and federated systems:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iifseclabel.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iifseclabel.html)

## Basic architecture with impersonation



Configuring IBM Db2 Big SQL security

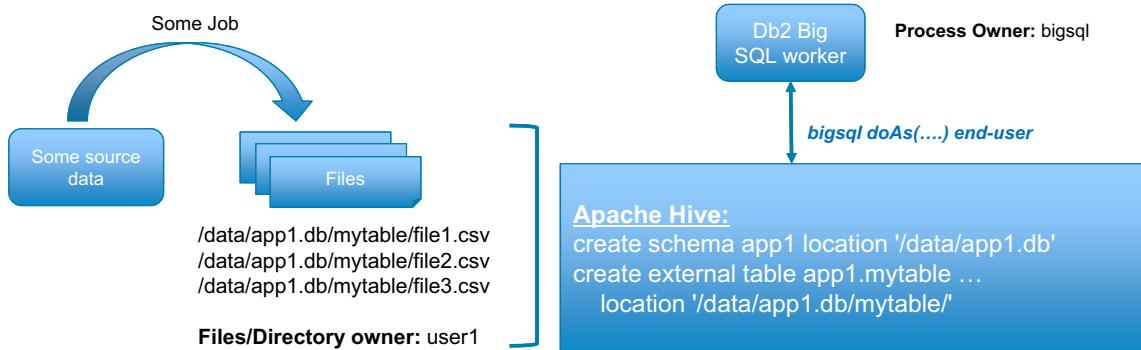
© Copyright IBM Corporation 2021

Figure 4-37. Basic architecture with impersonation

To understand the default security model of Db2 Big SQL, this slide illustrates what happens when different users, say User 1 and User 2, submit queries to Db2 Big SQL. The queries are compiled and optimized in the Db2 Big SQL head node, and instructions are passed to the workers. The workers are running as the `bigsq1` service ID. So, the files in HDFS are read by the `bigsq1` service ID on behalf of the user.



## Populating Apache Hive and Db2 Big SQL tables by using HDFS directly



- Files for `app1.mytable` on HDFS are owned by user1. For example, the files are the output of a Hadoop job that is run by user1.
- Any user can query `app1.mytable` if they have HDFS level permissions to `/data/app1.db/mytable`. Users or other application IDs that are external to Db2 Big SQL directly manage who can access their data sets.
- The `LOAD HADOOP` utility still works the same way, but any directories and files that are created are owned by the user in HDFS.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2020

Figure 4-38. Populating Apache Hive and Db2 Big SQL tables by using HDFS directly

This slide shows that user1 owns the `/data/app1.db/mytable/` directory. The directory contains files from a Hadoop job. To use Db2 Big SQL on this data, the bigsql service ID impersonates user1 and creates the schema along with the table, overlaying the data in Hadoop. Now, user1 can run SQL on the data in HDFS. The HDFS level permission of the `/data/app1.db/mytable` directory is set where any user can query that table. The `LOAD HADOOP` utility works the same way, but any new directory and files that are created are owned by the user1 in HDFS.



## Enabling impersonation for Db2 Big SQL

Verify that the `bigsq1` user is listed in the HDFS configuration property `hadoop.proxyusers.*`:

- `hadoop.proxyuser.<bigsq1>.groups`

This property should be set to one or more groups of users that the `bigsq1` user may impersonate.

- `hadoop.proxyuser.<bigsq1>.hosts`

This property should be set to a comma-separated list of all Big SQL head nodes and worker nodes (hosts) on which the `bigsq1` user may impersonate another user.

The screenshot shows the Ambari interface for managing HDFS configurations. The left sidebar lists services like Dashboard, Services, HDFS, YARN, MapReduce2, Tez, Hive, HBase, Pig, and Sqoop. The main area is titled 'Services / HDFS / Configs' and shows the 'CONFIGS' tab selected. A dropdown menu indicates 'Version: 3'. On the right, there are sections for 'SETTINGS' and 'ADVANCED'. Under 'Custom core-site', two properties are listed: 'hadoop.proxyuser.bigsq1.groups' and 'hadoop.proxyuser.bigsq1.hosts', each with an input field containing an asterisk (\*). The 'hadoop.proxy' config group is selected in the top right. The bottom right corner of the interface has a copyright notice: '© Copyright IBM Corporation 2021'.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-39. Enabling impersonation for Db2 Big SQL

The `bigsq1` user uses HDFS Secure Impersonation when it is impersonating another user. So, to verify whether you can enable impersonation for Db2 Big SQL, you must ensure that the `bigsq1` user is listed in the HDFS configuration property `hadoop.proxyusers.*`. This property allows the `bigsq1` user to impersonate other users. The property is added during the Big SQL installation only if the **Impersonation** checkbox is enabled during the Big SQL installation.

If impersonation is enabled after the Big SQL installation, then complete these steps to configure the properties:

1. From the Ambari dashboard HDFS service, click **Configs** and then the **Advanced** tab.
2. Expand the Custom core site to view the following properties:

- `hadoop.proxyuser.<bigsqI>.groups`

The default value is \*. Preferably, to be more restrictive, this property should be set to one or more groups of users that the bigsql user may impersonate. In this case, this list should include the group of the ambari-qa user, which is used in the Db2 Big SQL service check. Typically, this group is the “users” group.

- `hadoop.proxyuser.<bigsqI>.hosts`

The default value is \*. Preferably, to be more restrictive, this property should be set to a comma-separated list of all Big SQL head nodes and worker nodes (hosts) on which the bigsql user may impersonate another user.

There is a quicker way to do this verification by typing in the property name ‘`hadoop.proxyuser`’ in the Filter field, and then the properties appear. If the properties do not exist, you must create them to ensure that the bigsql user is listed in the HDFS configuration property. This action also allows the bigsql user to impersonate other users.

Also, Db2 Big SQL creates the impersonated table as the connected user in Apache Hive. It is assumed that the connected user has the appropriate authority to create a table in the specific schema. In addition, if the tables are created with impersonation and the `bigsqI.impersonation.create.table.grant.public` property is set to “true”, then the **INSERT**, **SELECT**, **DELETE**, and **UPDATE** operations are granted to public.

#### **Reference:**

Impersonation in Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big\\_sqI\\_impersonate.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.doc/doc/big_sqI_impersonate.html)

## Enabling impersonation in Apache Hive (1 of 2)

- If impersonation is enabled in Db2 Big SQL, all Apache Hive metadata operations and HDFS I/O operations are performed as the connected user.
- After turning on Apache Hive impersonation, you must set up permissions in HDFS.
- There are multiple suggestions to set up HDFS permissions, like changing the permissions level on the HDFS warehouse and disabling the `hive.warehouse.subdir.inherit.perms` property in Apache Hive:
  - Allow the `umask` setting of the current user account to dictate the permissions of directories and files that are created by them.
  - Set the value of `hive.warehouse.subdir.inherit.perms` property to `false`.
  - Set the HDFS `/apps/hive/warehouse` directory to 777 so that it is writable by all users.

*Figure 4-40. Enabling impersonation in Apache Hive (1 of 2)*

If impersonation is enabled in Db2 Big SQL, all Apache Hive metadata operations and HDFS I/O operations are performed as the connected user. To ensure that the connected user has appropriate permissions on HDFS, enable impersonation in Apache Hive too. Then, you have the proper authorizations that are configured in the HDFS for access through Apache Hive and Db2 Big SQL.

To turn on Apache Hive impersonation, complete the following steps:

1. From the Ambari web interface, click the Apache Hive service, and click the **Config** tab.
2. In the Filter field, search for `hive.server2.enable.doAs`.
3. A security tab appears with an On/Off button. Switch it to **On**.

Then, you must set up permissions in HDFS, which can be done multiple ways. One way is changing the permissions level on the HDFS warehouse and disabling the `hive.warehouse.subdir.inherit.perms` property in Apache Hive by completing the following steps:

1. Allow the “umask” setting of the current user account to dictate the permissions of directories and files that are created by them. You can verify or modify the `fs.permissions.umask-mode` from the Ambari web interface by selecting the HDFS service and clicking **Configs**, and then the **Advanced** tab. Then, expand the **Advanced hdfs-site** section. If the “umask” setting is updated, you must restart HDFS and possibly YARN.
2. Disable the `hive.warehouse.subdir.inherit.perms` property in the Apache Hive service so that new directories that are created under `/apps/hive/warehouse` do not inherit the permissions. Instead, you want to rely on the default UMASK:
  - a. Open the Apache Hive service and click **Configs**, and then the **Advanced** tab.
  - b. Expand the **Advanced hive-site** section.
  - c. Locate the `hive.warehouse.subdir.inherit.perms` property and set the value to “false”.
  - d. Click **Save**, and then restart the Apache Hive service.
3. Set the HDFS `/apps/hive/warehouse` directory to 777 so that it is writable by all users.

#### **Reference:**

Impersonation in Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsql\\_impersonate.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/bigsql_impersonate.html)

## Enabling impersonation in Apache Hive (2 of 2)

- When users create Apache Hive schemas, /apps/hive/warehouse must be writable by all users, as shown in the following example:  
`CREATE SCHEMA henry → /apps/hive/warehouse/henry.db`
- To allow users to create their own schemas, you must complete the following steps:
  - Set the HDFS /apps/hive/warehouse directory to 777 by running the following command:  
`hdfs dfs -chmod 777 /apps/hive/warehouse`

Path	ownership : group name	Permissions
/apps/hive/warehouse	hive:hadoop	rwxrwxrwx = 777

- Set the value of the `hive.warehouse.subdir.inherit.perms` property to `false`.

Figure 4-41. Enabling impersonation in Apache Hive (2 of 2)

You want to set the HDFS /apps/hive/warehouse directory to 777 so that all users can write to the directory.

For example, when you create a schema, a directory is created under the “warehouse” folder. To allow users to create their own schemas, you must set the inherit permissions properties to “false” along with setting the permissions to 777 on the “warehouse” directory.

## What is UMASK

- When users create Apache Hive schemas and tables, new directories are created:
  - CREATE SCHEMA henry → /apps/hive/warehouse/henry.db
  - CREATE TABLE henry.test (...) → /apps/hive/warehouse/henry.db/test/
- UMASK determines their permissions setting for new directories and files.
- What do you want it to be?

Start with*:	Subtract UMASK (022)
777 = rwxrwxrwx	$  \begin{array}{r}  777 \\  -022 \\  \hline  755  \end{array}  = \text{rwxr-xr-x}  $

\*Note: For files, start with 666 = -rw-rw-rw

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-42. What is UMASK

### So what is UMASK?

UMASK determines the permissions settings for new directories and files that are created. The default UMASK setting is 022, so when you start with 777, you subtract the UMASK value to get 755. This setting gives the owner full **READ**, **WRITE**, and **EXECUTE** permissions on the newly created schema. The group has **READ** and **EXECUTE**. All others have **READ** and **EXECUTE**.

## Examples

- Example 1: If user schema directories should have rwxr-xr-x (755), set UMASK=022 (default).

ownership : group name	Permissions	Path
user1:user1	rwxr-xr-x	/apps/hive/warehouse/user1
bigsq1:hadoop	rwxr-xr-x	/apps/hive/warehouse/user2

- Example 2: If user schema directories should have rwxrwx--- (770), set UMASK=007.

ownership : group name	Permissions	Path
user1:user1	rwxrwx---	/apps/hive/warehouse/user1
bigsq1:hadoop	rwxrwx---	/apps/hive/warehouse/user2

Figure 4-43. Examples

By adjusting the UMASK value to get what you need, you can control the permissions of the folders.

## Impersonation usage notes and restrictions

- If you enable the setting to retain HDFS trash, any **DROP** table statement automatically moves the table location to a **.Trash** folder under the home directory of the connected user in HDFS.
- If global impersonation is enabled, the user *username* can run **LOAD HADOOP** successfully only if the following items are true:
  - The HDFS directory `/user/<username>/` must exist and provide **READ**, **WRITE**, and **EXECUTE** permissions to the user.
  - The *<username>* user must exist on each node of the cluster.
- Any impersonation behavior is used for Hadoop tables only.
- Any storage handlers or SerDes that are used in a **CREATE HADOOP TABLE** statement also see the impersonation behavior.
- Impersonation is not used for tables that are created by the **CREATE HBASE TABLE** statement.

Figure 4-44. Impersonation usage notes and restrictions

### Usage notes

When you run a **DROP** table statement, Db2 Big SQL does the drop table operation in Apache Hive as the connected user if impersonation is enabled. Sometimes, Apache Hive allows the **DROP** to proceed with no errors even if an error exists when you delete the table location in HDFS. As a result, the data in HDFS is retained even if the table is dropped from Db2 Big SQL and Apache Hive. Make sure that you have the proper permissions as the connected user to perform the HDFS delete operation.

If you enable the setting to retain HDFS trash, any **DROP** table statement automatically moves the table location to a **.Trash** folder under the home directory of the connected user in HDFS. For this step to work, make sure that the HDFS directory `/user/<username>/` exists with **READ**, **WRITE**, and **EXECUTE** permissions for the user.

If global impersonation is enabled, the user *username* can run **LOAD HADOOP** successfully only if the following items are true:

- The HDFS directory `/user/<username>/` must exist and provide **READ**, **WRITE**, and **EXECUTE** permissions to the user.
- The *<username>* user must exist on each node of the cluster.

## Restrictions

- Any impersonation behavior is used for Hadoop tables only.
- Any storage handlers or SerDes that are used in a **CREATE HADOOP TABLE** statement also see the impersonation behavior.
- Impersonation is not used for tables that are created with the **CREATE HBASE TABLE** statement, even if the `bigsq1.alltables.io.doAs` property is set to “true”. For HBase tables, the `bigsq1` user creates a logical Db2 Big SQL table over an HBase table in Apache Hive in the `<schema>.db` directory in the Apache Hive warehouse. You must have appropriate permissions set up in HDFS for the `<schema>.db` directory. If the containing schema is created implicitly during a **CREATE HBASE TABLE** statement, it is owned by the `bigsq1` user. Any attempt to drop the schema explicitly is tried as the connected user, so ensure that there are appropriate permissions in HDFS for the connected user to perform the drop operation.

## Reference:

Impersonation in Db2 Big SQL:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq1.doc/doc/big\\_sq1\\_imperonate.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq1.doc/doc/big_sq1_imperonate.html)

## Unit summary

- Configured authentication for Db2 Big SQL.
- Managed security by using Apache Ranger.
- Enabled SSL encryption.
- Configured authorization of Db2 Big SQL objects.
- Configured impersonation in Db2 Big SQL.

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-45. Unit summary*

## Review questions

1. You can control authorization of Db2 Big SQL objects on which of the following levels?
  - A. With authorization in the distributed file system.
  - B. Authorization with the GRANT command.
  - C. Authorization at the row and column levels.
  - D. By using VIEWS or STORED PROCEDURES.
  - E. All the above.
  
2. True or False: You can Kerberize the cluster only after installing Db2 Big SQL.
  
3. True or False: RCAC controls access to a table at the row level, column level, or both.



Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-46. Review questions*

Write your answers here:

1. E. All the above
2. False. You can install Db2 Big SQL on a Kerberized cluster or Kerberize the cluster after installing Db2 Big SQL.
3. True.

## Review questions (cont.)

4. Which of the following items is true about Apache Ranger?
  - A. You must first install Apache Ranger, and then you can enable the Db2 Big SQL Apache Ranger plug-in
  - B. It is a framework to enable, monitor, and manage comprehensive data security across the Hadoop platform.
  - C. It can audit all access to Db2 Big SQL tables, views, and nicknames.
  - D. All the above.
  
5. True or False: UMASK is used to determine the permissions settings for new directories and files that are created.



Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-47. Review questions (cont.)

Write your answers here:

4. D. All the above.
5. True.

## Review answers

1. You can control authorization of Db2 Big SQL objects on which of the following levels?
  - A. With authorization in the distributed file system.
  - B. Authorization with the GRANT command.
  - C. Authorization at the row and column levels.
  - D. By using VIEWS or STORED PROCEDURES
  - E. All the above.
2. True or False: You can Kerberize the cluster only after installing Db2 Big SQL.
3. True or False: RCAC controls access to a table at the row level, column level, or both.



Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-48. Review answers

## Review answers (cont.)

4. Which of the following items is true about Apache Ranger?
  - A. You must first install Apache Ranger, and then you can enable the Db2 Big SQL Apache Ranger plug-in
  - B. It is a framework to enable, monitor, and manage comprehensive data security across the Hadoop platform.
  - C. It can audit all access to Db2 Big SQL tables, views, and nicknames.
  - D. All the above.
5. True or False: UMASK is used to determine the permissions settings for new directories and files that are created.



Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-49. Review answers (cont.)

## Exercise: Configuring authorizations of Db2 Big SQL objects

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-50. Exercise: Configuring authorizations of Db2 Big SQL objects*

## Exercise objectives



- Authorization of the Db2 Big SQL objects is controlled at several levels:
  - Level 1: Controlling access with authorization in the distributed file system.
  - Level 2: Authorization with the GRANT command.
  - Level 3: Authorization at the row and column level.
  - Level 4: Controlling access by using VIEWS or STORED PROCEDURES.

In this exercise, you focus on authorization at the row and column level.

- After completing this exercise, you will be able to:
  - Create roles, assign privileges to roles, and grant roles to users.
  - Use column masking and row-based access control to restrict access to your data.

## Exercise: Configuring impersonation in Db2 Big SQL

Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

*Figure 4-52. Exercise: Configuring impersonation in Db2 Big SQL*

## Exercise objectives

- Impersonation is the ability to allow a service user to securely access data in Hadoop on behalf of another user. In this exercise, you enable and configure impersonation in Db2 Big SQL
- After completing this exercise, you will be able to:
  - Configure impersonation in Db2 Big SQL.
  - Set up impersonation for non-admin users so that they can create a Hadoop table, load data into it, and run queries.



Configuring IBM Db2 Big SQL security

© Copyright IBM Corporation 2021

Figure 4-53. Exercise objectives

# Unit 5. Data federation with IBM Db2 Big SQL

## Estimated time

00:45

## Overview

In this unit, you learn about data federation with Db2 Big SQL.

## Unit objectives

- Explain the concept of Db2 Big SQL federation.
- List the supported data sources.
- Set up and configure a federated server to use different data sources.

## 5.1. Db2 Big SQL federation overview

## Db2 Big SQL federation overview

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-2. Db2 Big SQL federation overview*

## Topics

- Db2 Big SQL federation overview
  - Db2 Big SQL federation components

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-3. Topics*

## "Build it, and they will come."

- Hadoop vendors have long promoted the idea of centralizing all data into Hadoop to break down data silos:
  - Enterprise data hubs.
  - Data lakes.
- Many adopters of Hadoop started to ingest data from RDBMS sources to Hadoop:
  - Many or all tables, many sources, and all columns.
  - "Bring it all, just in case. It's cheap!"
- Although "cheaper", Hadoop is not free, so what happens?
  - Rapid cluster growth.
  - Data ingestion for all sources must be maintained.
  - Somebody must pay for it.

*Figure 5-4. "Build it, and they will come."*

For years, Hadoop vendors promoted Hadoop clusters to centralize all data into a single place to break down data silos. Because these projects are primarily IT-driven projects, there is a mentality of "build it and they will come", so many organizations started to build Hadoop clusters and the processes that are required to ingest the various data sources before business users asked for them.

In the past, data warehouses were expensive, and resources (such as disks and CPU) were relatively scarce. So, data sets were chosen carefully, and in some cases, columns were specifically picked to minimize the growth of the enterprise data warehouse.

With Hadoop, the practice is to ingest most or all tables from any source (prioritized by anticipated demand), and all columns are copied to keep things simple. Because Hadoop is a great extract, load, and transform (ELT) platform, copying all data makes sense because of the substantially lower cost of Hadoop.

Although Hadoop is "cheaper", it is not free, and Hadoop has the operational cost of maintaining and securing the platform.

The result? Many customers have substantial Hadoop clusters with much data (terabytes or even petabytes), but the business value or return on investment (ROI) is limited. Also, any new data sources require an operational process to bring that data into Hadoop and keep it current (for example, daily incremental ingest processes).

**References:**

- Hortonworks Data Platform:  
<https://www.ibm.com/downloads/cas/DKWR4KZB>
- IBM Db2 Big SQL:  
[https://www.ibm.com/products/db2-big-sql?lnk=STW\\_US\\_STESCH&lnk2=trial\\_Db2BigSQL&pxp=def&psrc=none&mhsrc=ibmsearch\\_a&mhq=Hadoop](https://www.ibm.com/products/db2-big-sql?lnk=STW_US_STESCH&lnk2=trial_Db2BigSQL&pxp=def&psrc=none&mhsrc=ibmsearch_a&mhq=Hadoop)
- ETL processing in Big SQL - Hadoop Dev:  
[https://www.ibm.com/support/pages/node/6259907?mhsrc=ibmsearch\\_a&mhq=ETL%20and%20Big%20SQL](https://www.ibm.com/support/pages/node/6259907?mhsrc=ibmsearch_a&mhq=ETL%20and%20Big%20SQL)

## Is there a better way?

- Value of the data is unknown initially.
- Users usually are not allowed to run ad hoc SQL statements on production data.
- What if the data was accessible, but only ingested to the data lake when the value of Hadoop is proven?

[Data federation with IBM Db2 Big SQL](#)

© Copyright IBM Corporation 2021

*Figure 5-5. Is there a better way?*

Business users often do not know what value is in their data sets until they see it. However, IT generally does not allow users to run ad hoc SQL statements against a production system. So, the users create processes to copy that data to a "safe place", like a data warehouse or Hadoop. After the data is available, they want to experiment with the data by visualizing it and joining it to other data sets, possibly from other systems. Therefore, "breaking down silos" is important, and for most people the only way to do this task is to load everything into Hadoop.

So is there a better way? For now, say that the customer has a simplified view where Hadoop is the data lake. What if you could provide access to the data but invest in ingest and maintenance processes only when the value was *proven*?

### Reference:

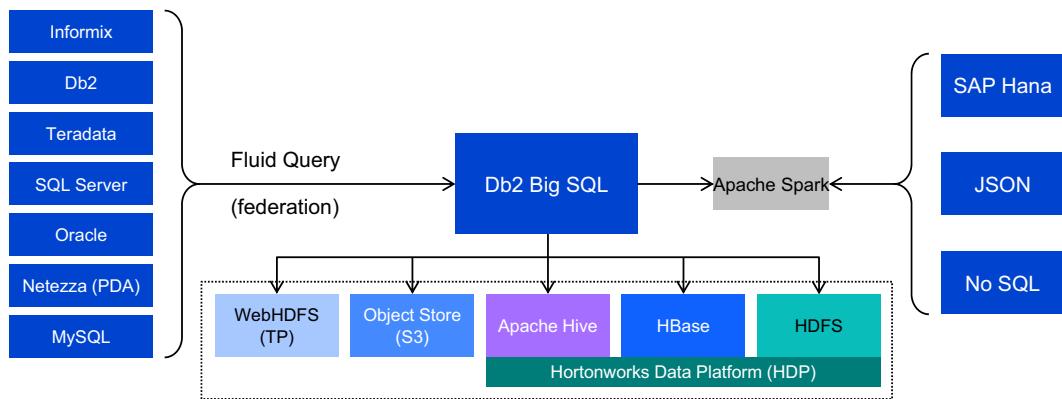
Big data integration and Hadoop:

<https://www.ibm.com/downloads/cas/JNY2OWKD>

## Solution: Db2 Big SQL with Fluid Query

Provided access to RDBMS data through a single connection to the data lake

- Robust ANSI SQL compliance
- Reduced cluster growth and sprawl
- Included (no additional cost) with Db2 Big SQL



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-6. Solution: Db2 Big SQL with Fluid Query

Db2 Big SQL has Fluid Query technology, which you can use to use leverage in Hadoop, but also join Hadoop data sets to data that is available in external databases. There are no special skills required: You just need a single connection to Db2 Big SQL.

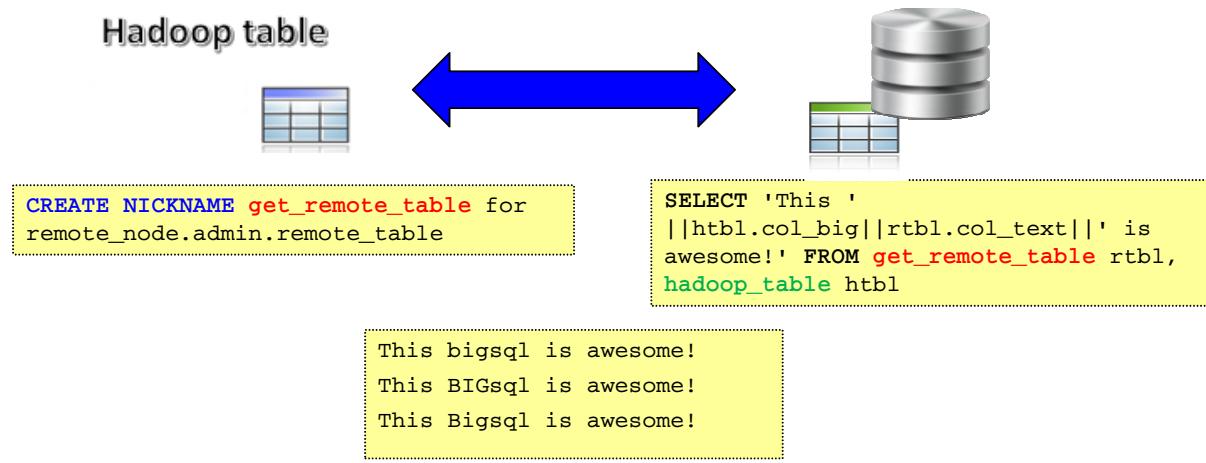
### References:

Federation capability:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview\\_federation.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqldoc/doc/overview_federation.html)

## Db2 Big SQL federation overview

- Users can process SQL queries and statements from multiple data sources:
  - The queries and statements act as ordinary tables or views.
  - You can integrate existing data sources with Db2 Big SQL, such as Db2 LUW, Oracle, Teradata, and Netezza (IBM PureData for Analytics).
- Federation is one of the key features and differentiators of Db2 Big SQL.



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-7. Db2 Big SQL federation overview

One of the key differentiators of Db2 Big SQL is its ability to process queries and statements from multiple data sources as though they were ordinary tables or views. A user can integrate existing data sources with Db2 Big SQL without needing to migrate the data into Db2 Big SQL. Some of the data sources that are currently supported are listed in this slide, and the next few slides cover the details.

Here is an example of how federation works in Db2 Big SQL:

1. Create a nickname for a remote table.
2. Select from that nickname and other Db2 Big SQL tables as though it was a table within Db2 Big SQL. The results that are returned are consolidated.

**References:**

- Data sources that are supported by the federation that is bundled in Big SQL V6.0.0:  
<https://www.ibm.com/support/pages/node/870650>
- Nicknames:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfcdsma01.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfcdsma01.html)
- The **CREATE NICKNAME** statement:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002169.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002169.html)

## Federated system

- A special type of distributed database management system (DBMS) that consists of the following items:
  - An instance that operates as a federated server.
  - A database (the default is *bigsq1*) that acts as the federated database.
  - One or more data sources.
  - Clients (users and applications) that access the database and data sources.
- Characteristics:
  - *Transparent*: Appears to be one source.
  - *Extensible*: Brings together data sources.
  - *Autonomous*: There is no interruption to data sources, applications, and systems.
  - *High functioning*: Has full query support against all data (such as scalar functions and stored procedures).
  - *High performance*: Has optimization of distributed queries.

[Data federation with IBM Db2 Big SQL](#)

© Copyright IBM Corporation 2021

Figure 5-8. Federated system

In a federated system, you have an instance that operates as a federated server. You also have a federated database (the default database is *bigsq1*). You have one or more data sources. You have your clients who are the users, the applications that access the database, and the data sources.

Here are the characteristics of a federated system:

- *Transparent*: Appears to be one source.
- *Extensible*: Brings together data sources.
- *Autonomous*: There is no interruption to data sources, applications, and systems.
- *High functioning*: Has full query support against all data (such as scalar functions and stored procedures).
- *High performance*: Has optimization of distributed queries.

**References:**

- Federation capability:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../overview\\_federation.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../overview_federation.html)

- Default privileges that are granted on the *bigsq*l database:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../bi\\_admin\\_bigsq\\_defpriv.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../bi_admin_bigsq_defpriv.html)

- Scalar function:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/c0000767.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.../commsq...l.doc/doc/c0000767.html)

- Federated stored procedure:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfointfsp.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfointfsp.html)

## Supported data sources

Db2 Big SQL supports the following data sources:

- IBM Db2
- Oracle
- Oracle MySQL
- Teradata
- IBM PureData System for Analytics (formerly Netezza)
- Microsoft SQL Server
- Apache Spark SQL
- CouchDB
- IBM MQ
- Microsoft Azure SQL Database
- MongoDB
- Parquet on Hadoop
- SAP Sybase

[Data federation with IBM Db2 Big SQL](#)

© Copyright IBM Corporation 2021

*Figure 5-9. Supported data sources*

Here are examples of the data sources that are supported by the federation feature that is bundled in Db2 Big SQL V6.0.0:

- IBM Db2
- Oracle
- Oracle MySQL
- Teradata
- IBM PureData System for Analytics (formerly Netezza)
- Microsoft SQL Server
- Apache Spark SQL
- CouchDB
- IBM MQ
- Microsoft Azure SQL Database
- MongoDB
- Parquet on Hadoop
- SAP Sybase

**Reference:**

Data sources that are supported by the federation feature that is bundled in Db2 Big SQL V6.0.0:

<https://www.ibm.com/support/pages/node/870650>

## Example: SELECT

What the client sees or submits:

```
SELECT c.cust_nation, sum(o.totalprice)
FROM customer c, orders o
WHERE o.orderstatus = 'OPEN'
    and c.custkey=o.custkey
    and c.mktsegment = 'BUILDING'
GROUP BY c.cust_nation
```



What the federated server does:

Joins rows from both sources. Sorts them by `cust_nation` and sums the total order price for each nation. Returns the result to the application.



```
SELECT o.custkey
FROM orders o
WHERE o.orderstatus = 'OPEN'
```



```
SELECT c.custkey, c.cust_nation
FROM customer c
WHERE c.mktsegment = 'BUILDING'
```

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-10. Example: SELECT

Here is an example of Db2 Big SQL in action. In the query at the top of the slide, the query looks like any ordinary query. You cannot tell that the data from the two selected tables that are selected is from different data sources. Db2 Big SQL combines the results from both data sources and groups them before returning the data to the application. This transparency makes it easy to work with multiple data sources.

### Reference:

Selecting data in a federated system – examples:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/rfpwrk04.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/rfpwrk04.html)

## Federated system main components

- The federated server
- The federated database
- Wrappers
- Server definitions
- User mappings
- Nicknames and data source objects

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-11. Federated system main components

### The federated server

The database server in a federated system is referred to as the *federated server*. Any number of database instances can be configured to function as federated servers.

The database instance that manages the federated system is called a server because it responds to requests from users and client applications. The federated server often sends parts of the requests that it receives to the data sources for processing. A *pushdown operation* is an operation that is processed remotely. The database instance that manages the federated system is referred to as the *federated server*, even though it acts as a client when it pushes down requests to the data sources.

### The federated database

To users and client applications, data sources appear as a single collective database in the database system. Users and applications interface with the *federated database* that is managed by the federated server.

The federated database contains a system catalog that stores information about data. The federated database system catalog contains entries that identify data sources and their characteristics. The federated server reviews the information that is stored in the federated database system catalog and the data source wrapper to determine the best plan for processing SQL statements.

The next slides illustrate the other components in detail.

**Reference:**

Federated system:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint01.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint01.html)

## 5.2. Db2 Big SQL federation components

## Db2 Big SQL federation components

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-12. Db2 Big SQL federation components*

## Topics

- Db2 Big SQL federation overview
- ▶ Db2 Big SQL federation components

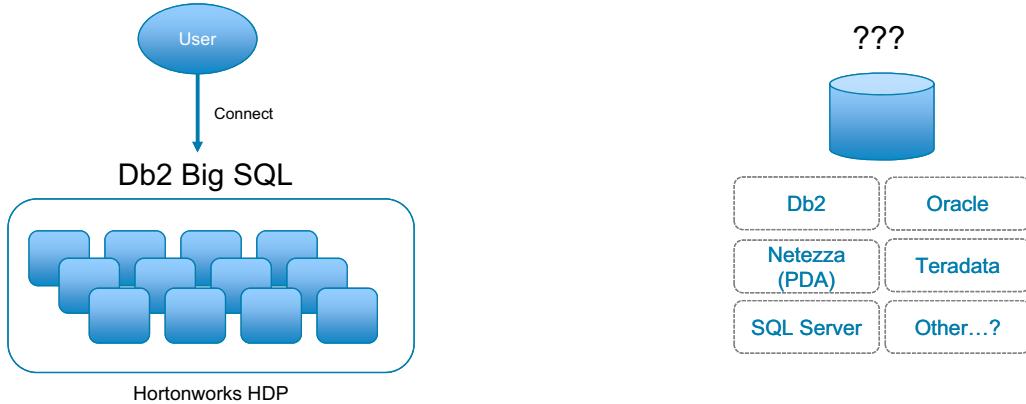
Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-13. Topics*

## Federation components (1 of 5)

- A user connects to a Db2 Big SQL cluster running on Hortonworks Data Platform.
- The user wants to connect to a remote data source, which might be any of the supported ones.

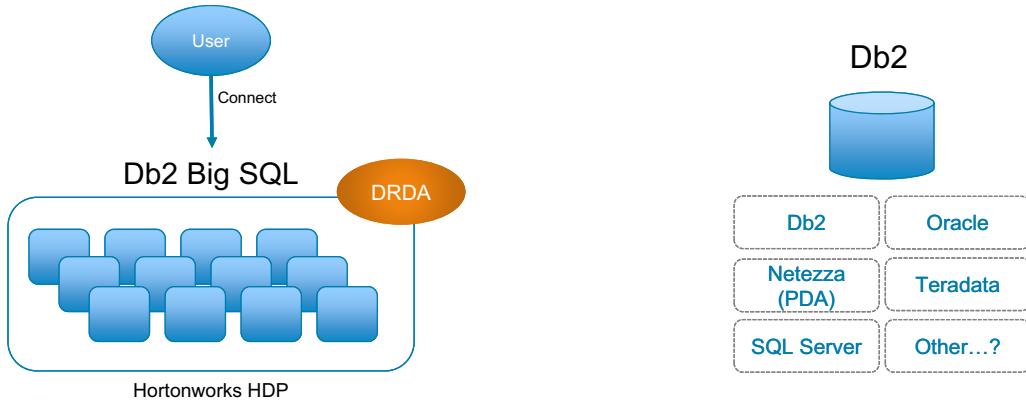


*Figure 5-14. Federation components (1 of 5)*

A user connects to the Db2 Big SQL cluster running on Hortonworks Data Platform and wants to connect to a remote data source that has valuable data, which might be any of the supported ones on the list.

## Federation components (2 of 5): Wrappers

- *Wrappers* identify the protocols to use for the remote RDBMS.
- Db2 Big SQL is an “application client” to the remote RDBMS.



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-15. Federation components (2 of 5): Wrappers

To connect to any database, you must know what the remote system type is, which leads to using the correct database driver and protocol. In Db2 Big SQL, the protocol is called a *wrapper*.

The federated server uses the wrapper to communicate with and retrieve data from the data sources, where each data source has its own wrapper.

Db2 Big SQL Fluid Query supports many types of RDBMS sources, including the ones that you see here.

A wrapper is implemented as a set of library files. The default wrapper name for the Db2 family data sources is *DRDA*. When you use the default name to register the wrapper, you do not need to specify the library name because the federated server automatically uses the default library name that is associated with the wrapper.

**References:**

- Wrappers and wrapper modules:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint07.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint07.html)

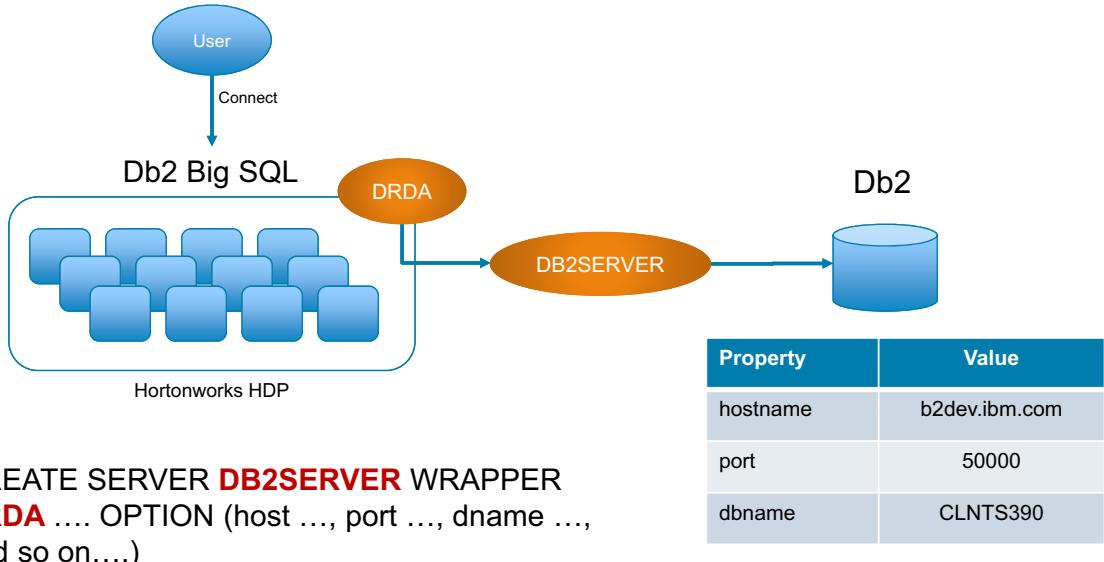
- Configuring access to Db2 data sources:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/tlsdb201.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/tlsdb201.html)

- Registering the Db2 wrapper:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/tlsdb204.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/tlsdb204.html)

## Federation components (3 of 5): Server definitions



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-16. Federation components (3 of 5): Server definitions

After wrappers are created for the data sources, the federated instance owner defines the data sources to the federated database.

You must know how to find the remote database. To do so, use a SERVER object. When you create the server, it uses a default wrapper for that server type and Db2 Big SQL knows how to reach the remote database. Run a **CREATE SERVER** statement while connected to Db2 Big SQL, and those connection properties are stored in a SERVER object.

The instance owner supplies a name to identify the data source and other information that pertains to the data source. This information includes the following items:

- The type and version of the data source
- The database name for the data source (RDBMS only)
- Metadata that is specific to the data source, like host name and port number

The name and other information that the instance owner supplies to the federated server are collectively called a *server definition*.

The example that is shown in the slide is for a Db2 database. A Db2 database family data source can have multiple databases. The definition must specify which database the federated server can connect to. In contrast, an Oracle data source has one database, and the federated server can connect to the database without knowing its name. The database name is not included in the federated server definition of an Oracle data source.

Data sources answer requests for data and are servers too.

The **CREATE SERVER** and **ALTER SERVER** statements are used to create and modify a server definition.

The following example shows you how to register a server definition for a DRDA wrapper by using the **CREATE SERVER** statement:

```
CREATE SERVER DB2SERVER TYPE DB2/LUW VERSION 11 WRAPPER DRDA  
AUTHORIZATION "spalten" PASSWORD "db2guru" OPTIONS (HOST 'db2dev.ibm.com', PORT  
'50000', DBNAME 'CLNTS390')
```

#### References:

- Server definitions and server options:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint08.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint08.html)

- **CREATE SERVER** statement:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002170.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsq.commsql.doc/doc/r0002170.html)

- **CREATE SERVER** statement - Examples for the Db2 wrapper:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/rlsdb206.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/rlsdb206.html)

## Federation components (3 of 5): Server definitions (cont'd)

- A server defines the properties and options of a specific data source:
  - Type and version
  - Database name for the data source (assuming RDBMS only)
  - Other metadata
- The server is defined by using the **CREATE SERVER** statement:
  - A generic wrapper for this type of data source is created when the **CREATE SERVER** statement runs.

Figure 5-17. Federation components (3 of 5): Server definitions (cont'd)

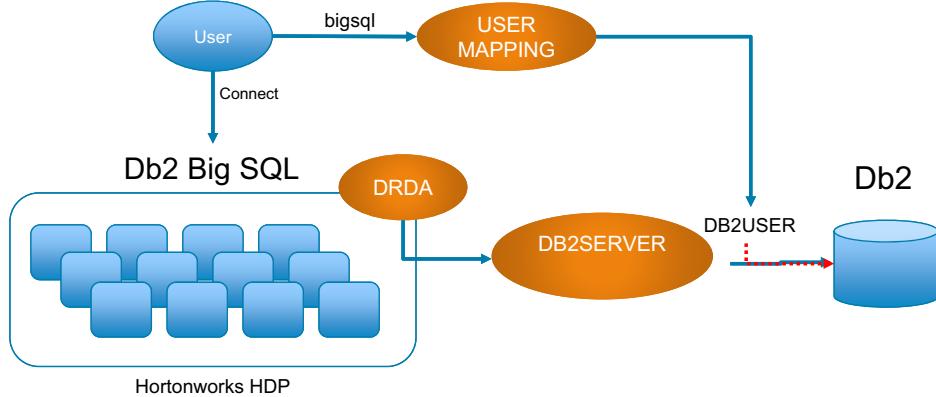
The server acts as the medium between the application and the data source. You define the properties and the options of specific data source. The server is defined by using the **CREATE SERVER** statement. A generic wrapper is created for the server type when you issue the **CREATE SERVER** statement.

### Reference:

Server definitions and server options:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint08.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint08.html)

## Federation components (4 of 5): User mappings



```

CREATE USER MAPPING FOR bigsq
 SERVER DB2SERVER
 OPTION (remote_auth_id ..., remote_password ...)

```

Figure 5-18. Federation components (4 of 5): User mappings

A *user mapping* is an association between an authorization ID on the federated server and the information that is required to connect to the remote data source.

To create a user mapping, you specify the following information in the **CREATE USER MAPPING** statement:

- Local authorization ID
- Local name of the remote data source server as specified in the server definition
- Remote ID and password

When the user issues an SQL statement to connect to the remote server, the federated server performs these steps:

1. Retrieves the user mapping that is related to that user.
2. Decrypts the remote password *remote\_pw* that is associated with the remote server.
3. Calls the wrapper to connect to the remote server.
4. Passes the remote ID *remote\_ID* and the decrypted remote password to the wrapper.
5. Creates a connection to the remote server for that user.

After the server is set up, you must be able to handle a situation where the credentials that are required to access the remote tables might be different from the users that access the *bigsq1* database, or perhaps you want to create specialized "read only" connection credentials for this use case in general.

For example, the remote source might create a user ID that is subject to workload management throttling and the workload is identified by this special ID that Db2 Big SQL uses. This setup protects the source system from well-intended but potentially resource-expensive data requests.

For example, assume that the user connection from Db2 Big SQL is called *bigsq1*, but the ID to connect to is *DB2USER*. You want this mapping to happen automatically, which is established by using a **CREATE USER MAPPING** statement. Now, *bigsq1* always is mapped as *DB2USER* whenever access to the remote data source is attempted.

**Reference:**

User mappings:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint11.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint11.html)

## Federation components (4 of 5): User mappings (cont'd)

An association between an authorization ID on the federated server and the information that is required to connect to the remote data source can be defined by using the **USER** special register. For example:

```
CREATE USER MAPPING FOR USER ...
```

Figure 5-19. Federation components (4 of 5): User mappings (cont'd)

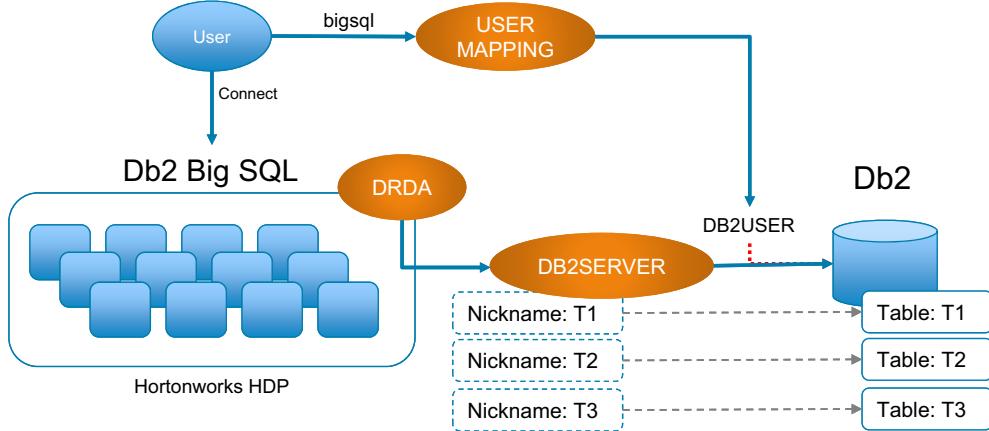
A user mapping is an association between the authorization ID and the information that is required to connect to the remote data source. This step might be optional depending on how the data source is set up. If the environment uses federated trusted contexts and proxy authentication, none or only a few user mappings might be required. You can define the user by using the **USER** special register or a specific user such as "Henry" or **PUBLIC**.

### Reference:

User mappings:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint11.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint11.html)

## Federation components (5 of 5): Nicknames



**CREATE NICKNAME ... FOR ...**

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-20. Federation components (5 of 5): Nicknames

A **nickname** is an identifier that you use to identify the data source object, like tables that you want to access in the remote data source.

A nickname is not an alternative name for a data source object like an alias is an alternative name. A nickname is the pointer by which the federated server references the object.

Nicknames are typically defined with the **CREATE NICKNAME** statement along with specific nickname column options and nickname options.

When a client application or a user submits a distributed request to the federated server, the request does not need to specify the data sources. Instead, the request references the data source objects by their nicknames. The nicknames are mapped to specific objects at the data source. These mappings eliminate the need to qualify the nicknames by data source names. The location of the data source objects is transparent to the client application or the user.

The Db2 Big SQL administrator has full control over which tables from the remote system are exposed to users in Hadoop. To make a remote table accessible, use the **CREATE NICKNAME** statement. Nicknames look like tables in Db2 Big SQL but are pointers to objects in the remote system. Also, as more tables from multiple systems are virtualized under Hadoop, the more the chance of table name collisions increases. Nicknames provide you with full control to rename objects or schemas as necessary to best organize your data sets under one virtualized environment.

**Reference:**

Nicknames and data source objects:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint12.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/cfpint12.html)

## Federated three-part names

- No need to define a nickname for each object that is accessed.
- Zero administration impact:
  - Schema changes are “picked up” with ease.
  - Simply specify:

```
SELECT ... FROM <server_name>.<schema>.<table name>
```

```
SELECT .... FROM ORACLE.STAGING.COST_ANALYSIS_1992
```

- Security can be handled through user mappings.

*Figure 5-21. Federated three-part names*

In some cases, there might be hundreds of objects in the remote system. Creating nicknames for each of those objects has a high impact. In addition, if the schema or anything changes, all the nicknames must be updated.

Federated three-part names enable you to directly access the remote object with zero impact. You specify `<server_name>.<schema>.<table_name>`, where metadata for remote objects that are referenced in federated three-part names is retrieved at run time. This feature improves the synchronization of locally stored metadata with remote servers.

Federated three-part table names are supported for all relational data sources.

### References:

- Federated three-part names:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfq3pnintro.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfq3pnintro.html)
- Support and considerations for federated three-part names:  
[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfq3pnsupport.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.data.fluidquery.doc/topics/iiyfq3pnsupport.html)

## Unit summary

- Explained the concept of Db2 Big SQL federation.
- Listed the supported data sources.
- Set up and configured a federated server to use different data sources.

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-22. Unit summary*

## Review questions

1. What are the characteristics of a federated system?
  - A. Transparent
  - B. Extensible
  - C. Autonomous
  - D. High performance
  - E. High function
  - F. All the above
  
2. Which of the following products are supported data sources?
  - A. Db2
  - B. Oracle
  - C. Teradata
  - D. MS SQL Server
  - E. All the above
  
3. True or False: User mappings are used to authenticate to the remote data source.

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-23. Review questions

Write your answers here:

1. F. All the above.
2. E. All the above.
3. True.



## Review questions (cont.)

4. True or False: The federated database manages the federated system and often sends parts of the requests it receives to the data sources for processing.
  
5. Which of the following items is responsible for defining the data sources to the federated database, besides defining the property and values of the connection?
  - A. Wrappers
  - B. Server definitions
  - C. User mappings
  - D. Nicknames
  
6. True or False: Nicknames can be used for wrappers and servers.



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-24. Review questions (cont.)

Write your answers here:

4. False. The federated server manages the federated system and often sends parts of the requests it receives to the data sources for processing.
5. B. Server definitions.
6. False. Nicknames are used for tables or views.

## Review answers

1. What are the characteristics of a federated system?
  - A. Transparent
  - B. Extensible
  - C. Autonomous
  - D. High performance
  - E. High function
  - F. All the above
2. Which of the following products are supported data sources?
  - A. Db2
  - B. Oracle
  - C. Teradata
  - D. MS SQL Server
  - E. All the above
3. True or False: User mappings are used to authenticate to the remote data source.



Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

Figure 5-25. Review answers

## Review answers (cont.)

4. True or False: The federated database manages the federated system and often sends parts of the requests it receives to the data sources for processing.
  
5. Which of the following items is responsible for defining the data sources to the federated database, besides defining the property and values of the connection?
  - A. Wrappers
  - B. Server definitions
  - C. User mappings
  - D. Nicknames
  
6. True or False: Nicknames can be used for wrappers and servers.



## Exercise: Getting started with Db2 Big SQL federation

Data federation with IBM Db2 Big SQL

© Copyright IBM Corporation 2021

*Figure 5-27. Exercise: Getting started with Db2 Big SQL federation*

## Exercise objectives

- Within a federated system, a single SQL statement can access data that is distributed among one or more data sources. In this exercise, you follow the basic steps to get started with the federation feature of Db2 Big SQL.
- After completing this exercise, you should be able to:
  - Create a Db2 server instance on IBM Cloud.
  - Create a server object to connect your federated Db2 Big SQL server to the remote data source.
  - Create user mapping objects.
  - Create a nickname.
  - Run queries from the federated Db2 Big SQL server to query the data on the Db2 service instance on IBM Cloud.





IBM Training



© Copyright International Business Machines Corporation 2016, 2021.

Course Exercises Guide

# **Big Data Engineer 2021**

## ***IBM Db2 Big SQL***

Course code SABSQ ERC 3.0

Ahmed Abdel-Baky

Maria Farid

Heba Aboulmagd

Abdelrahman Hassan

Mohamed El-Khouly

Norhan Khaled

Adel El-Metwally

Ramy Said

Nouran El-Sheikh

Dina Sayed



## February 2021 edition

### Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

### Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

**© Copyright International Business Machines Corporation 2016, 2021.**

**This document may not be reproduced in whole or in part without the prior written permission of IBM.**

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Trademarks .....</b>	<b>v</b>
<b>Exercises description .....</b>	<b>vi</b>
<b>Information for instructors .....</b>	<b>vii</b>
<b>Exercise 1. Connecting to the IBM Db2 Big SQL server .....</b>	<b>1-1</b>
Part 1: Adding the Hadoop cluster hostname to the hosts file .....	1-4
Part 2: Checking services status from the Ambari Web UI .....	1-6
Part 3: Connecting to the Db2 Big SQL server by using JSqsh .....	1-8
Part 4: Exploring the JSqsh documentation with the help command .....	1-11
Part 5: Running basic Db2 Big SQL statements by using JSqsh .....	1-12
Part 6: Exploring Db2 Big SQL through Ambari by using the Db2 Big SQL console .....	1-19
<b>Exercise 2. Creating and managing Db2 Big SQL schemas and tables .....</b>	<b>2-1</b>
Part 1: Creating and dropping a simple Db2 Big SQL table .....	2-2
Part 2: Creating sample tables .....	2-4
Part 3: Moving data into HDFS .....	2-12
Part 4: Loading data into Db2 Big SQL tables .....	2-14
Part 5: Creating and working with views .....	2-18
Part 6: Populating a table with 'INSERT INTO ... SELECT' .....	2-19
Part 7: Working with external tables .....	2-20
<b>Exercise 3. Querying Db2 Big SQL tables .....</b>	<b>3-1</b>
Part 1: Connecting to Db2 Big SQL .....	3-3
Part 2: Querying data with Db2 Big SQL .....	3-4
Part 3: Working with the ARRAY type .....	3-7
Part 4: Working with Db2 Big SQL functions .....	3-10
Part 5: Storing data in an alternative file format (Parquet) .....	3-13
<b>Exercise 4. Configuring authorizations of Db2 Big SQL objects .....</b>	<b>4-1</b>
Part 1: Setting up the scenario .....	4-3
Part 2: Creating roles and assigning privileges to the roles .....	4-7
Part 3: Creating and enforcing the permissions .....	4-8
Part 4: Granting roles to users and activating the row access control .....	4-10
Part 5: Switching between users in Db2 Big SQL Console .....	4-11
Part 6: Using row access control .....	4-11
Part 7: Shielding data with column masking .....	4-13
<b>Exercise 5. Configuring impersonation in Db2 Big SQL .....</b>	<b>5-1</b>
Part 1: Enabling impersonation in Db2 Big SQL .....	5-3
Part 2: Setting the permissions to the warehouse directory on HDFS .....	5-6
Part 3: Copying the sample data to HDFS .....	5-7
Part 4: Creating a Hadoop table, loading data, and querying the table .....	5-7
<b>Exercise 6. Getting started with Db2 Big SQL federation .....</b>	<b>6-1</b>
Part 1: Preparing the remote data source for this exercise .....	6-3
Part 2: Creating a server object .....	6-12
Part 3: Creating the user mapping object .....	6-14

Part 4: Creating the nickname .....	6-15
Part 5: Running queries .....	6-15

---

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

Db2®

IBM Cloud™

Linux® is a registered trademark used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies.

# Exercises description

This course includes the following exercises:

- Exercise 1. Connecting to the IBM Db2 Big SQL server

In this exercise, you connect to the Db2 Big SQL server by using various techniques. First, you explore the lab environment. Then, you learn how to set up the Java SQL shell (JSqsh) and use it to connect to the Db2 Big SQL server. Finally, you explore the Db2 Big SQL service by using the Db2 Big SQL console graphical web interface.

- Exercise 2. Creating and managing Db2 Big SQL schemas and tables

In this exercise, you start by creating and dropping a simple Hadoop table by using Db2 Big SQL. Then, you create multiple tables by using various data types and load the tables with data. You also work with views, external tables, and other methods of creating Db2 Big SQL tables.

- Exercise 3. Querying Db2 Big SQL tables

In this exercise, you experiment with advanced SQL queries. Then, you explore the Db2 Big SQL ARRAY type. You also create a user-defined function (UDF) and write queries that call the UDF. Finally, you store data in an alternative file format (Parquet).

- Exercise 4. Configuring authorizations of Db2 Big SQL objects

Authorization of the Db2 Big SQL objects is controlled at several levels. In this exercise, you focus on authorization at the row and column level.

- Exercise 5. Configuring impersonation in Db2 Big SQL

Impersonation is the ability to allow a service user to securely access data in Hadoop on behalf of another user. In this exercise, you enable and configure impersonation in Db2 Big SQL.

- Exercise 6. Getting started with Db2 Big SQL federation

Within a federated system, a single SQL statement can access data that is distributed among one or more data sources. In this exercise, you follow the basic steps to get started with the federation feature of Db2 Big SQL.

# Information for instructors

Instructors must complete the following steps *before* students start the exercises in this course:

- \_\_\_ 1. Confirm that all students completed the exercises in the course "Big Data Ecosystem".
- \_\_\_ 2. Obtain the following values from the IBM Skills Academy administrator:
  - Ambari admin <**ambari admin**>
  - Ambari admin password <**ambari admin pwd**>
  - Db2 Big SQL password <**bigsq1\_password**>
  - root password <**root\_password**>
- \_\_\_ 3. Complete the steps described in "[Start services for Db2 Big SQL course](#)".
- \_\_\_ 4. Wait until all students complete Exercises 1, 2, and 3.
- \_\_\_ 5. Before students start Exercise 6, complete the steps that are described in "[Steps to run before Big SQL Exercise 6](#)"



## Reminder

Exercises 4 and 5 require administration authority that is not granted to the students in this course. These exercises can be performed only by the instructor at the end of the course. A recorded demo of these exercises is available from the IBM Skills Academy portal.

## Troubleshooting

For up-to-date troubleshooting information and tips, see "[Troubleshooting](#)".

---

# Exercise 1. Connecting to the IBM Db2 Big SQL server

## Estimated time

01:00

## Overview

In this exercise, you connect to the Db2 Big SQL server by using various techniques. First, you explore the lab environment. Then, you learn how to set up the Java™ SQL shell (JSqsh) and use it to connect to the Db2 Big SQL server. Finally, you explore the Db2 Big SQL service by using the Db2 Big SQL console graphical web interface.

## Objectives

After completing this exercise, you will be able to:

- Use the Ambari Web UI to check the status of various services, including Db2 Big SQL.
- Connect to a Db2 Big SQL server and run Db2 Big SQL queries from the JSqsh shell.
- Run Db2 Big SQL statements by using the JSqsh shell.
- Launch the Db2 Big SQL console from the Ambari Web UI.
- Run SQL statements from the Db2 Big SQL console.
- Display several monitoring capabilities from the Db2 Big SQL console.

## Introduction

IBM Db2 Big SQL is a hybrid SQL on Hadoop engine. It delivers advanced, scalable, and security-rich data querying for the enterprise business. Db2 Big SQL delivers massive parallel processing (MPP) and advanced data query. Db2 Big SQL offers a single database connection or query for disparate sources such as Hadoop HDFS and WebHDFS, RDMS, NoSQL databases and object stores. It provides low latency, high performance, security, SQL compatibility and federation capabilities to do ad hoc and complex queries.

Data scientists and analysts can reuse their SQL skills, saving the time spent in retraining.

The Db2 Big SQL server or service consists of one Db2 Big SQL head (two heads in an HA configuration) that is installed on a node that is called the head node, and multiple Db2 Big SQL workers that are installed on nodes that are called worker nodes.

You can access Db2 Big SQL through the following methods:

- Java SQL Shell (JSqsh)
- Db2 Big SQL console
- Any client tool that supports IBM JDBC or ODBC driver

## Requirements

Copy the following table to your favorite text editor (for example Notepad) and assign the values that are provided to you by your instructor to the variables in this table.



### Note

The variables listed under “**INSTRUCTOR ONLY**” are used by the instructor and *not* shared with the students.

*Table 1. Variables and values for the students and the instructor in this course*

Variable	Value	Description
Username	<username>	Student's username
Password	<password>	Student's password
IP address	<ip_address>	Hadoop cluster & VM IP address
Host name	<hostname>	Hadoop cluster host name
Ambari URL	<hostname:8080>	URL to access Ambari Web UI
Ambari username	<ambari username>	Username to access Ambari Web UI
Ambari password	<ambari password>	Password to access Ambari Web UI
Cluster name	<cluster_name>	Hadoop cluster name
HDFS home dir	<hdfs_home>	HDFS home directory. The value for the exercises is /user/<username>
User account home dir	\$HOME	Account home directory. The value is /home/<username>. It does not need to be substituted in the exercises.

### **INSTRUCTOR ONLY**

Ambari admin	<ambari admin>	Ambari administrator
Ambari admin password	<ambari admin pwd>	Ambari administrator password
Db2 Big SQL password	<bigsq1_password>	<b>bigsq1</b> user password
root password	<root_password>	root user password



### Note

The hostnames that are shown in the screen captures of this exercise might be different from the ones in your hands-on environment.

- PuTTY SSH client installed in your workstation.

## Exercise instructions

In this exercise, you complete the following tasks:

- \_\_\_ 1. Add the Hadoop cluster hostname to the hosts file.
- \_\_\_ 2. Check services status from the Ambari Web UI.
- \_\_\_ 3. Connect to the Db2 Big SQL server by using JSqsh.
- \_\_\_ 4. Explore the JSqsh documentation with the help command.
- \_\_\_ 5. Run basic Db2 Big SQL statements by using JSqsh.
- \_\_\_ 6. Explore Db2 Big SQL through Ambari by using the Db2 Big SQL console.

### **Part 1: Adding the Hadoop cluster hostname to the hosts file**

The Hadoop cluster hostname and domain are not added to a DNS. To resolve the hostname from the IP address, you must add an entry to the hosts file. Update the hostfile with the <hostname> and <ip\_address> provided by your instructor. Follow the instructions for your workstation operating system to update the hosts file.



#### Note

Skip this part if you already added the same hostname to your hosts file in the Big Data Ecosystem course, “Exercise 1. Exploring the lab environment”.

For Windows, the hosts file is at C:\Windows\System32\drivers\etc\hosts. Follow these steps for Windows 10 workstations:

- \_\_\_ 1. Press the **Windows** key.
- \_\_\_ 2. Type **Notepad** in the search field.
- \_\_\_ 3. In the search results, right-click Notepad and select **Run as administrator**.
- \_\_\_ 4. From Notepad, open the following file:  
c:\Windows\System32\Drivers\etc\hosts
- \_\_\_ 5. Add the values for <ip\_address> and <hostname> in the last line of the file. For example,  
10.1.1.1 dataengineer.ibm.com
- \_\_\_ 6. Click **File > Save** to save your changes.
- \_\_\_ 7. Ping the VM by using the <hostname> that you configured in the hosts file. For example,  
ping dataengineer.ibm.com
- \_\_\_ 8. You receive the response from the hosts as shown in the following figure.

```
C:\>ping dataengineer.ibm.com

Pinging dataengineer.ibm.com [REDACTED] with 32 bytes of data:
Reply from [REDACTED]: bytes=32 time=263ms TTL=48
Reply from [REDACTED]: bytes=32 time=275ms TTL=48
Reply from [REDACTED]: bytes=32 time=256ms TTL=48
Reply from [REDACTED]: bytes=32 time=258ms TTL=48

Ping statistics for [REDACTED]:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 256ms, Maximum = 275ms, Average = 263ms
```

For Linux, the hosts file is in the directory `/etc/hosts`. Follow these steps for Linux computers:

- \_\_\_ 1. Open a Linux terminal. You must be logged in as `root` or as a user who can run `sudo`.  
Add the values for `<ip_address>` and `<hostname>` in the hosts file:  
`sudo echo <ip_address> <hostname> >> /etc/hosts`  
Example:  
`sudo echo 10.1.1.1 dataengineer.ibm.com >> /etc/hosts`
- \_\_\_ 2. Ping the VM by using the `<hostname>` that you configured in the hosts file. For example,  
`ping dataengineer.ibm.com`  
You receive the response from the host as shown in the following figure.

```
[root@d9035236bcfd /]# ping dataengineer.ibm.com
PING dataengineer.ibm.com (10.1.1.1) 56(84) bytes of data.
64 bytes from dataengineer.ibm.com (10.1.1.1): icmp_seq=1 ttl=37 time=20.1 ms
64 bytes from dataengineer.ibm.com (10.1.1.1): icmp_seq=2 ttl=37 time=14.6 ms
64 bytes from dataengineer.ibm.com (10.1.1.1): icmp_seq=3 ttl=37 time=13.8 ms
^C
--- dataengineer.ibm.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 13.859/16.235/20.178/2.807 ms
[root@d9035236bcfd /]# █
```

For Mac, the hosts file is in the directory `/private/etc/hosts`. Follow these steps for Mac computers:

- \_\_\_ 1. Open a Mac terminal. You must be logged in as `root` or as a user who can run `sudo`.  
Edit the hosts file:  
`sudo nano /private/etc/hosts`  
Add the values for `<ip_address>` and `<hostname>` in the last line of the file. For example,  
`10.1.1.1 dataengineer.ibm.com`
- \_\_\_ 2. Save the changes by typing `^X` (Control + X) to exist, then type `Y` if you are prompted to confirm to save the file. Press `Enter` to confirm the filename.

- \_\_\_ 3. Ping the VM by using the <hostname> that you configured in the hosts file. For example,  
ping dataengineer.ibm.com

You receive the response from the hosts as shown in the following figure.

```
[MacBook-Pro:~ adelnour$ ping dataengineer.ibm.com
PING dataengineer.ibm.com (10.1.1.1): 56 data bytes
64 bytes from 10.1.1.1: icmp_seq=0 ttl=59 time=14.153 ms
64 bytes from 10.1.1.1: icmp_seq=1 ttl=59 time=17.592 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=59 time=26.617 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=59 time=13.174 ms
^C
--- dataengineer.ibm.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 13.174/17.884/26.617/5.302 ms
```

## **Part 2: *Checking services status from the Ambari Web UI***

You can install Db2 Big SQL on a Hortonworks Data Platform (HDP). Then, you can enable the Db2 Big SQL extension on HDP so that Db2 Big SQL appears in the list of services that you can add in Ambari. You can start, stop, and check the status of the Db2 Big SQL server from Ambari.

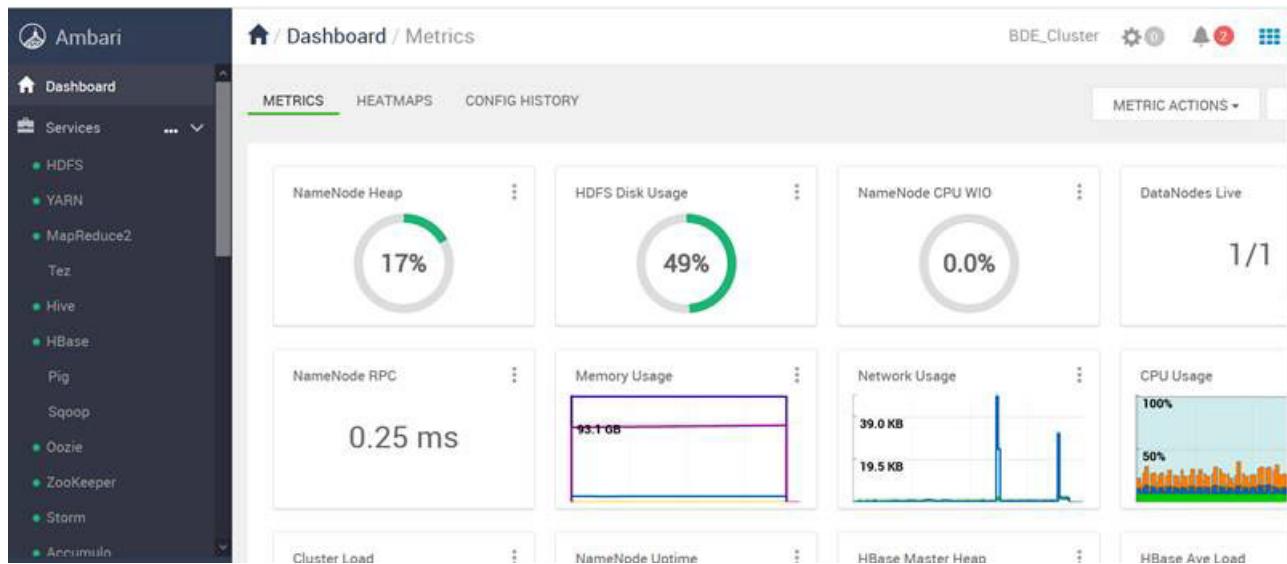


### Note

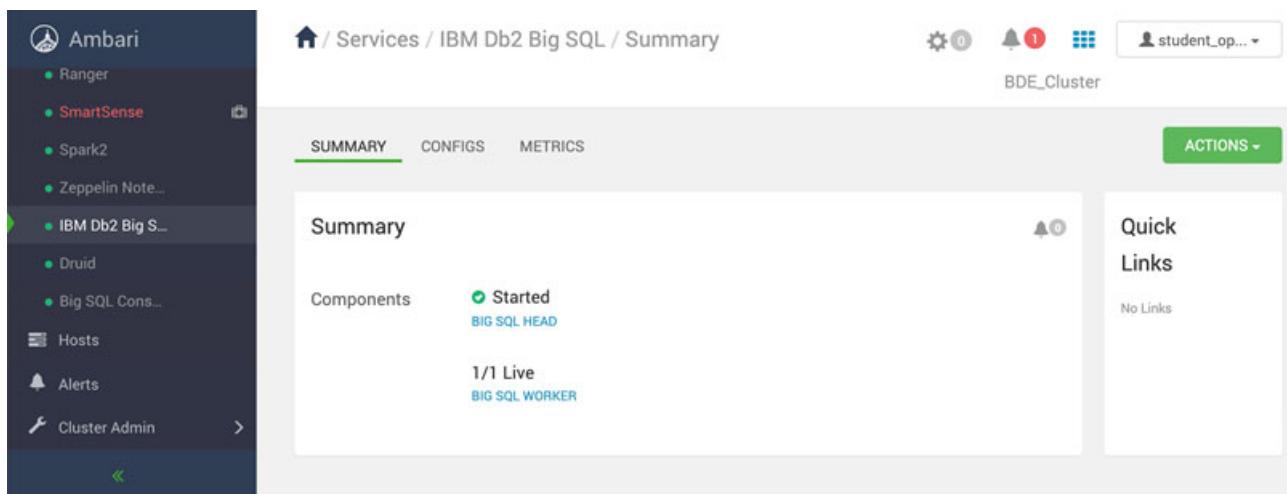
In this course, you do not have authority to start and stop services.

Complete the following steps:

- \_\_\_ 1. Enter the Ambari URL <hostname:8080> in your browser address bar.
- \_\_\_ 2. Sign in to the Ambari Web UI with your Username <ambari\_username> and Password <ambari\_password>.
- \_\_\_ 3. Expand **Services**. The services are listed on the left pane.



- \_\_\_ 4. Scroll down and select the **IBM Db2 Big SQL** service.
- \_\_\_ 5. In the SUMMARY tab, confirm that all the components are started.



- \_\_\_ 6. Click **Big SQL Console** and confirm that the components are started.

The screenshot shows the 'SUMMARY' tab selected in the top navigation bar. The main content area is titled 'Summary'. Under 'Components', there is a status entry for 'BIG SQL CONSOLE' which is marked as 'Started' with a green checkmark.

### **Part 3: Connecting to the Db2 Big SQL server by using JSqsh**

Db2 Big SQL queries are run by the Db2 Big SQL server on your cluster against data in your cluster. To run Db2 Big SQL queries, your environment must be associated with a Db2 Big SQL server that is part of the Db2 Big SQL service. To run Db2 Big SQL queries, you must first connect to a Db2 Big SQL server. You can run Db2 Big SQL queries from JSqsh.

In this part, you start the JSqsh shell and set up the connection information for Db2 Big SQL.

- 1. Open **PuTTY** to connect to the VM. Enter the <hostname>, <username>, and <password>.
- 2. Start the JSqsh shell by running the following command:  

```
/usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh
```

If it is your first time opening JSqsh, the WELCOME TO JSQSH! Screen is displayed.
- 3. If the "WELCOME TO JSQSH!" screen is displayed, press **Enter** to walk through this welcome screen and then enter **C** to start the **Connection wizard**.

```
[bigsql@dataengineer ~]$ /usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh
Welcome to JSqsh 6.0.0.3
Type "\help" for help topics. Using JLine.
WELCOME TO JSQSH!
```

It looks like this is the first time that you've run jsqsh (or you just typed '\help welcome' at the jsqsh prompt). If this is the first time you have run jsqsh, you will find that you have a shiny new directory called '.jsqsh' in your home directory and that this directory contains a couple of files that you should be aware of:

**drivers.xml** - JSqsh comes pre-defined to understand how to use a fixed number of JDBC drivers, however this file may be used to teach it how to recognize other JDBC drivers. The file is pretty well commented, so hopefully it is enough to get you started.

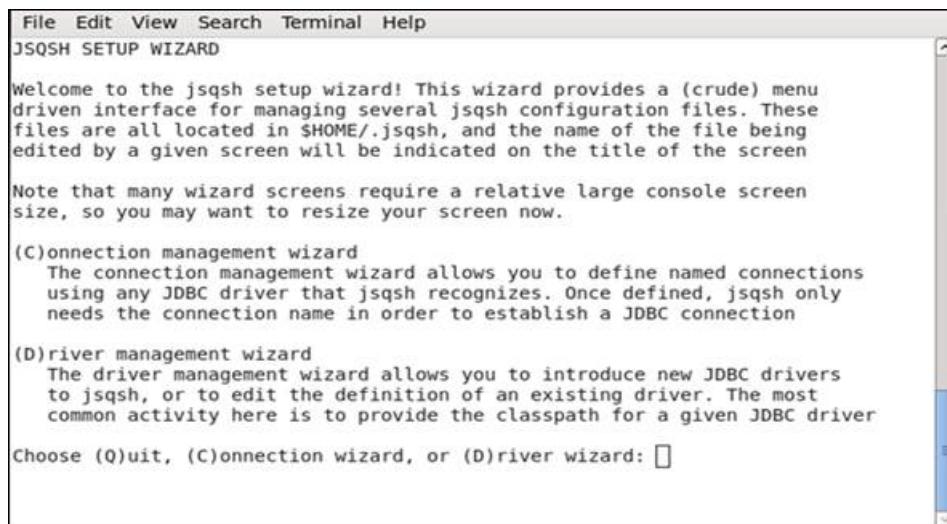
**sqshrc** - Everything contained in this file is executed just as if you had typed commands at the sqsh prompt and is the place where you can set variables and configure sqsh to your likings.

JSqsh is intended to be self-documenting. If you would like to see this information again, then type '\help welcome' at the jsqsh prompt, or run '\help' to see a list of all help topics that are available.

You will now enter the jsqsh setup wizard.  
Hit enter to continue: █

If this is not your first time opening JSqsh, "WELCOME TO JSQSH!" is not displayed. Instead, a prompt like "1>" is displayed. You can now enter input commands. If you are at this prompt, you enter a command to open the JSqsh Setup Wizard. Enter the following command to open the Setup Wizard:

\setup



Once you see the **JSQSH SETUP WIZARD** screen, enter **C** to start the connection wizard.

- \_\_\_ 4. Enter **1** to select the **bigsq1** connection:

```
JSQSH CONNECTION WIZARD - (edits $HOME/.jsqsh/connections.xml)
The following connections are currently defined:
```

Name	Driver	Host	Port
1 bigsql	db2	dataengineer.ibm.com	32051

Enter a connection number above to edit the connection, or:  
 (B)ack, (Q)uit, or (A)dd connection: █

- \_\_\_ 5. For each of the Connection URL Variables, ensure that your values match the ones that are displayed. The following figure provides an example. “server” should match your <hostname>, “user” should match your <username>.

JSQSH CONNECTION WIZARD - (edits \$HOME/.jsqsh/connections.xml)

The following configuration properties are supported by this driver.

```
Connection name : bigsql
  Driver : IBM Data Server (DB2, Informix, Big SQL)
  JDBC URL : jdbc:db2://${server}:${port}/${db}
```

#### Connection URL Variables

```
-----
1      db : BIGSQL
2      port : 32051
3      server : dataengineer.ibm.com
4      user : student0000
5      password :
6      Autoconnect : false
```

#### JDBC Driver Properties

```
-----
None
```

Enter a number to change a given configuration property, or  
 (T)est, (D)elete, (B)ack, (Q)uit, Add (P)roperty, or (S)ave: █

- \_\_\_ 6. Enter a password in the “password” variable. Enter **5** to select the fifth variable, “password”, and type your password <**password**>.
- \_\_\_ 7. Enter **T** to test the connection.

```
Enter a number to change a given configuration property, or
(T)est, (D)elete, (B)ack, (Q)uit, Add (P)roperty, or (S)ave: T

Attempting connection...
Succeeded!

Hit enter to continue: ■
```

If your connection is properly configured, the message "Succeeded!" message is displayed.

- \_\_\_ 8. Enter **S** to save your configurations.
- \_\_\_ 9. Enter **Y** to confirm to save the password.
- \_\_\_ 10. Enter **Q** to quit.

At this point, you have successfully connected to the **bigsq1** database and schemas.

#### **Part 4: Exploring the JSqsh documentation with the help command**

To learn more about the syntax and use of the JSqsh command, use the **\help** command. More online documentation is available at <http://sourceforge.net/apps/mediawiki/jsqsh/>. To get more help on a particular topic, type **\help** with the topic of interest, for example:

```
\help commands
\help vars
```

In this part, you become familiar with JSqsh **\help**.

Complete the following steps:

- \_\_\_ 1. From the JSqsh shell, type **\help** to display a list of available help categories.

```
1> \help
Available help categories. Use "\help <category>" to display topics within that
category
+-----+
| Category | Description
+-----+
| commands | Help on all available commands
| vars     | Help on all available configuration variables
| topics   | General help topics for jsqsh
+-----+
1> ■
```

- \_\_\_ 2. Type **\help commands** to display help for the supported commands.

A partial list of supported commands is displayed on the initial screen.

1> \help commands
Available commands. Use "\help <command>" to display detailed help for a given command
+-----+-----+
Command   Description
+-----+-----+
\alias        Creates an alias
\buf-append   Appends the contents of one SQL buffer into another
\buf-copy      Copies the contents of one SQL buffer into another
\buf-edit      Edits a SQL buffer
\buf-load      Loads an external file into a SQL buffer
\call          Call a prepared statement
\connect       Establishes a connection to a database.
\create        Generates a CREATE TABLE using table definitions
\databases     Displays set of available databases (catalogs)
\debug         (Internal) Used to enable debugging
\describe      Displays a description of a database object
\diff          Compares results from multiple sessions
\drivers       Displays a list of JDBC drivers known by jsqsh.
\echo          Displays a line of text.
\end           Ends the current session
\eval          Read and execute an input file full of SQL
\globals       Displays all global variables
\go            Executes the contents of the current buffer
--More--

Press the **space bar** to display the next page or **q** to quit the display of help information.

## Part 5: Running basic Db2 Big SQL statements by using JSqsh

When you run the JSqsh command from the command line, you open a command shell. You can type specific Db2 Big SQL commands or statements into this shell and view output from Big SQL queries.

In this section, run simple JSqsh commands and Db2 Big SQL queries to become familiar with the JSqsh shell.

- 1. From the **JSqsh** shell, connect to your Db2 Big SQL server by using the connection **bigsq1** that you created in a previous task.

\connect bigsql

If prompted for a password, enter your password <**password**>.

- 2. Display essential information about all available tables one page at a time. Run the following command.

\show tables -e | more

If you're working with a newly installed Db2 Big SQL server, your results are similar to the following output.

```
1> \connect bigsql
[dataengineer.ibm.com] [student0000] 1> \show tables -e | more
+-----+-----+-----+
| TABLE_CAT | TABLE_SCHEM | TABLE_NAME | TABLE_TYPE |
+-----+-----+-----+
+-----+-----+-----+
[dataengineer.ibm.com] [student0000] 1>
```

\_\_ 3. Select the first 5 rows from SYSCAT.TABLES (the Db2 Big SQL catalog schema):

```
select tabschema, tablename from syscat.tables fetch first 5 rows only;
```

```
[dataengineer.ibm.com] [bigsql] 1>      select tabschema, tablename from syscat.ta
s fetch first 5 rows only;
+-----+-----+
| TABSCHEMA | TABNAME   |
+-----+-----+
| BIGSQL    | SMOKE_HADOOP2_1525422353 |
| BIGSQL    | SMOKE_HADOOP2_1639502892 |
| IBMCONSOLE | ALERT      |
| IBMCONSOLE | ALERT_PROPERTIES |
| IBMCONSOLE | BLOCKING_LOG_ALERTS |
+-----+-----+
5 rows in results(first row: 0.024s; total: 0.025s)
[dataengineer.ibm.com] [bigsql] 1> █
```



### Note

Your output will be different depending on your class environment.

When you work with large volumes of data, a useful development technique is to restrict the number of rows returned by a query.

Next, you review the history of commands that you recently ran in the JSqsh shell.

\_\_ 4. Type `\history` and press **Enter**.

Notice that the statements that were previously run are prefixed with a number in parentheses. You can reference this number in the JSqsh shell to recall that query.

```
[dataengineer.ibm.com] [bigsql] 1> \history
(1) select tabschema, tablename from syscat.tables
(2) select tabschema, tablename from syscat.tables fetch first 5 rows only
```

\_\_ 5. Type `!!` (two exclamation marks, without spaces) to recall the previously run statement.

The previous statement selects the first 5 rows from SYSCAT.TABLES.

```
[dataengineer.ibm.com] [bigsql] 1> !!
[dataengineer.ibm.com] [bigsql] 1> select tabschema, tablename from syscat.tables
fetch first 5 rows only
```

- \_\_\_ 6. To run the statement, type a semi-colon on the following line.

```
[dataengineer.ibm.com] [bigsq] 1> !!
[dataengineer.ibm.com] [bigsq] 1> select tabschema, tablename from syscat.tables
fetch first 5 rows only
[dataengineer.ibm.com] [bigsq] 2> ;
+-----+-----+
| TABSCHEMA | TABNAME      |
+-----+-----+
| IBMCONSOLE | CONTENTION_CONNECTION |
| IBMCONSOLE | DATABASE      |
| IBMCONSOLE | DBSTATUS       |
| IBMCONSOLE | DBSUMMARYSESSIONS   |
| IBMCONSOLE | DB_PROFILE_LINK    |
+-----+-----+
5 rows in results(first row: 0.032s; total: 0.034s)
```

Recall a previous SQL statement by referencing the number reported via the \history command.

For example, if you wanted to recall the 4th statement, you would type !4. In the previous example, only two statements were run.

After the statement is recalled, add a semicolon (;) to the final line to run the statement.

Next, you experiment with the ability of JSqsh to support piping of output to an external program.

- \_\_\_ 7. Enter the following two lines on the command shell:

```
select tabschema, tablename from syscat.tables
go | more
```

The `go` statement in the second line causes the query on the first line to be run.

Note that there is no semicolon at the end of the SQL query on the first line. The semi-colon is a Db2 Big SQL short cut for the JSqsh `go` command.

The `| more` clause causes the output that results from running the query to be piped through the Unix/Linux `more` command to display one screen of content at a time.

Your results are similar to the following output.

```
[dataengineer.ibm.com] [bigsq] 1> select tabschema, tablename from syscat.tables
[dataengineer.ibm.com] [bigsq] 2> go | more
+-----+-----+
| TABSCHEMA | TABNAME
+-----+-----+
| IBMCONSOLE | MESSAGESAPSHOT
| IBMCONSOLE | MONITOR_PROFILE
| IBMCONSOLE | OBJECT_STORE
| IBMCONSOLE | REAL_MEMORY_STATS
| IBMCONSOLE | RESOURCE_CPU
| IBMCONSOLE | RESOURCE_LOGSPACE
| IBMCONSOLE | RESOURCE_MEMORY
| IBMCONSOLE | RESOURCE_STORAGEGROUP
| IBMCONSOLE | RTMON_MAP_COL
| IBMCONSOLE | RTMON_MAP_DBCCONN
| IBMCONSOLE | RTMON_MAP_QUERY
| IBMCONSOLE | RTMON_MAP_STMT_TYPE
| IBMCONSOLE | RTMON_MAP_SUBSET
| IBMCONSOLE | VIRTUAL_MEMORY_STATS
| IBMCONSOLE | WORKLOAD
| IBMCONSOLE | bottleCpu
| IBMCONSOLE | bottleDirectReads
| IBMCONSOLE | bottleDirectWrites
| IBMCONSOLE | bottleElapsedTime
| IBMCONSOLE | bottleFcmRw
--More--780 rows in results(first row: 0.006s; total: 0.084s)
```

- \_\_\_ 8. Since there are hundreds of rows to display in this example, enter **q** to quit displaying further results and return to the JSqsh shell.

Next, experiment with JSqsh's ability to redirect output to a local file rather than the console display.

- \_\_\_ 9. Enter the following lines in the command shell. Adjust the path information on the final line as needed for your environment:

```
select tabschema, colname, colno, typename, length
from syscat.columns
fetch first 50 rows only
go > $HOME/test1.out
```

In this example, the output of the query in the first line is directed to the output file **test1.out** in your user's account home directory (/home/<username>).

The following figure shows the results of this step.

```
[dataengineer.ibm.com] [student0001] 1> select tabschema, colname, colno, typename, length
[dataengineer.ibm.com] [student0001] 2> from syscat.columns
[dataengineer.ibm.com] [student0001] 3> fetch first 50 rows only
[dataengineer.ibm.com] [student0001] 4> go > $HOME/test1.out
50 rows in results(first row: 0.063s; total: 0.085s)
```

- \_\_\_ 10. Exit the shell:

```
quit
```

\_\_\_ 11. In your PuTTY session, view the output file:

```
cat $HOME/test1.out
```

The result is similar to the following output.

TABSCHEMA	COLNAME	COLNO	TYPENAME	LENGTH
SYSIBM	NAME	0	VARCHAR	128
SYSIBM	CREATOR	1	VARCHAR	128
SYSIBM	TYPE	2	CHARACTER	1
SYSIBM	CTIME	3	TIMESTAMP	10
SYSIBM	REMARKS	4	VARCHAR	254
SYSIBM	PACKED_DESC	5	BLOB	133169152
SYSIBM	VIEW_DESC	6	BLOB	4190000
SYSIBM	COLCOUNT	7	SMALLINT	2
SYSIBM	FID	8	SMALLINT	2
SYSIBM	TID	9	SMALLINT	2
SYSIBM	CARD	10	BIGINT	8
SYSIBM	NPAGES	11	BIGINT	8
SYSIBM	FPAGES	12	BIGINT	8
SYSIBM	OVERFLOW	13	BIGINT	8
SYSIBM	PARENTS	14	SMALLINT	2
SYSIBM	CHILDREN	15	SMALLINT	2
SYSIBM	SELFREFS	16	SMALLINT	2
SYSIBM	KEYCOLUMNS	17	SMALLINT	2
SYSIBM	KEYOBID	18	SMALLINT	2
SYSIBM	REL_DESC	19	BLOB	262144
SYSIBM	BASE_NAME	20	VARCHAR	128
SYSIBM	BASE_SCHEMA	21	VARCHAR	128
SYSIBM	TBSpace	22	VARCHAR	128
SYSIBM	INDEX_TBSpace	23	VARCHAR	128
SYSIBM	LONG_TBSpace	24	VARCHAR	128
SYSIBM	KEYUNIQUE	25	SMALLINT	2
SYSIBM	CHECKCOUNT	26	SMALLINT	2
SYSIBM	CHECK_DESC	27	BLOB	131072
SYSIBM	STATS_TIME	28	TIMESTAMP	10
SYSIBM	DEFINER	29	VARCHAR	128
SYSIBM	TRIG_DESC	30	BLOB	3145728
SYSIBM	DATA_CAPTURE	31	CHARACTER	1
SYSIBM	STATUS	32	CHARACTER	1
SYSIBM	CONST_CHECKED	33	CHARACTER	32
SYSIBM	PMAP_ID	34	SMALLINT	2
SYSIBM	ENCODING_SCHEME	35	CHARACTER	1
SYSIBM	PCTFREE	36	SMALLINT	2
SYSIBM	ROWTYPESCHEMA	37	VARCHAR	128
SYSIBM	ROWTYPEPENAME	38	VARCHAR	128
SYSIBM	APPEND_MODE	39	CHARACTER	1
SYSIBM	PARTITION_MODE	40	CHARACTER	1
SYSIBM	REFRESH	41	CHARACTER	1
SYSIBM	REFRESH_TIME	42	TIMESTAMP	10

SYSIBM	LOCKSIZE	43	CHARACTER	1
SYSIBM	VOLATILE	44	CHARACTER	1
SYSIBM	REMOTE_DESC	45	BLOB	10485760
SYSIBM	CLUSTERED	46	CHARACTER	1
SYSIBM	AST_DESC	47	BLOB	535822336
SYSIBM	DROPRULE	48	CHARACTER	1
SYSIBM	LOGINDEXBUILD	49	CHARACTER	1

Next, you invoke JSqsh by using an input file that contains the Db2 Big SQL commands to run. Maintaining SQL script files can be handy to repeatedly run various queries.

From the Unix/Linux command line, use any available editor to create a new file in your local directory named **test.sql**.

- \_\_\_ 12. Change into your home directory:

```
cd $HOME
```

- \_\_\_ 13. Type:

```
vi test.sql
```

- \_\_\_ a. Enter **Insert** mode by pressing "i"on your keyboard, and add the following 2 queries into your file:

```
select tabschema, tablename from syscat.tables fetch first 5 rows only;
select tabschema, colname, colno, typename, length
from syscat.columns
fetch first 10 rows only;
```

- \_\_\_ b. Save your file. Press the **Esc** key to exit INSERT mode, then type :wq and then press **Enter** to go back to the command line.



### Note

If Esc from PuTTY does not work, press **Ctrl + C** instead.

- \_\_\_ c. Invoke JSqsh and specify to connect to your Db2 Big SQL database and run the contents of the script that you previously created.

```
/usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh bigsql < test.sql
```

In this example, **bigsql** is the name of the database connection that you created in a previous task.

- \_\_\_ d. Inspect the output.

JSqsh executes each instruction and displays its output. The following figure shows partial results.

```
[bigsq1@dataengineer ~]$ /usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh bigsql < test.sql
Welcome to JSqsh 6.0.0.3
Type "\help" for help topics. Using JLine.
[dataengineer.ibm.com] [bigsq1] 1> select tabschema, tablename from syscat.tables fetch first 5 rows only;
+-----+-----+
| TABSCHEMA | TABNAME
+-----+-----+
| BIGSQL   | SMOKE_HADOOP2_1525422353 |
| BIGSQL   | SMOKE_HADOOP2_1639502892 |
| IBMCONSOLE | ALERT
| IBMCONSOLE | ALERT_PROPERTIES
| IBMCONSOLE | BLOCKING_LOG_ALERTS
+-----+
5 rows in results(first row: 0.041s; total: 0.045s)
[dataengineer.ibm.com] [bigsq1] 1> select tabschema, colname, colno, typename, length from syscat.columns fetch first 10 rows only;
+-----+-----+-----+-----+
| TABSCHEMA | COLNAME | COLNO | TYPENAME | LENGTH
+-----+-----+-----+-----+
| SYSIBM   | NAME      | 0     | VARCHAR   | 128
| SYSIBM   | CREATOR    | 1     | VARCHAR   | 128
| SYSIBM   | TYPE       | 2     | CHARACTER | 1
| SYSIBM   | CTIME      | 3     | TIMESTAMP | 10
| SYSIBM   | REMARKS    | 4     | VARCHAR   | 254
| SYSIBM   | PACKED_DESC | 5     | BLOB      | 133169152
| SYSIBM   | VIEW_DESC   | 6     | BLOB      | 4190000
| SYSIBM   | COLCOUNT   | 7     | SMALLINT  | 2
| SYSIBM   | FID        | 8     | SMALLINT  | 2
| SYSIBM   | TID        | 9     | SMALLINT  | 2
+-----+
10 rows in results(first row: 0.020s; total: 0.029s)
[dataengineer.ibm.com] [bigsq1] 1>
[dataengineer.ibm.com] [bigsq1] 2> [bigsq1@dataengineer ~]$ █
```

- \_\_\_ 14. Exit the Linux shell.

```
exit
```

## **Part 6: Exploring Db2 Big SQL through Ambari by using the Db2 Big SQL console**

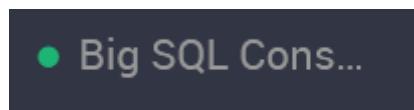
You can use the Db2® Big SQL console for database monitoring, administration, and configuration, and to run SQL queries. The console also provides historical monitoring data and key health indicators such as availability, responsiveness, and throughput.

You can launch the Db2 Big SQL console by using the quick link in the Ambari Web UI.

In this part, you start the Db2 Big SQL console from the Ambari Web UI to explore several options in the console. You display several monitoring options and run SQL statements.

Complete the following steps:

- \_\_\_ 1. Open the Ambari Web UI in your browser. Enter the Ambari URL <hostname:8080>. Log in with the <ambari user> and the <ambari password> that were provided to you by your instructor.
- \_\_\_ 2. Click **Big SQL Console** in the service menu.



- \_\_\_ 3. Select **Console UI** from the **Quick Links** menu.

The screenshot shows the 'SUMMARY' tab selected in the top navigation bar. Below it, a 'Components' section lists one item: 'BIG SQL CONSOLE' with a status of 'Started'. To the right, a 'Quick Links' panel contains a single link labeled 'Console UI'.

If prompted, login to **Db2 Big SQL** console by using your student account <username> and <password>.



### Note

If you receive the error “We can’t connect to the server at dataengineer.ibm.com” or similar, replace the hostname in the URL by the IP address of the VM that was provided by your instructor.

- 4. From the menu at the upper left (hamburger icon), click **CONNECTION INFO -> Connection Information**.

The screenshot shows the 'IBM Db2 Big SQL' interface with a sidebar menu. The 'CONNECTION INFO' section is expanded, and the 'Connection Information' sub-item is selected. On the right, there's a summary card for 'DATABASE RESPONSIVENESS' showing 'Last 1 hour' data: 5.4k Statements total in last 1 hour, with a histogram for execution time from 0ms-10ms.

- 5. Review the configured **CONNECTION INFORMATION**.

IBM Db2 Big SQL

**CONNECTION INFORMATION**

Connect your apps and clients to IBM Db2 Big SQL

#### Connection configuration resources

##### Without SSL

**Host name:** dataengineer.ibm.com  
**Port number:** 32051  
**Database name:** bigsql  
**User ID:** student0000  
**Password:** \*\*\*\*\*  
**Version:** Compatible with Db2, Version 11.1.0 or later

#### JDBC string

```
jdbc:db2://dataengineer.ibm.com:32051/bigsql:user=student0000;password=<your_password>;securityMechanism=3;
```

[Copy JDBC String](#)

#### More information

- [Db2 driver package \(IBM Knowledge Center\)](#)
- [IBM Data Server Client Packages](#)
- [Connecting CLPPlus to a Db2 database \(IBM Knowledge Center\)](#)

6. From the menu at the upper left, click **Monitor -> Overview** to check the summary of the Db2 Big SQL service health.

IBM Db2 Big SQL

**EXPLORE**

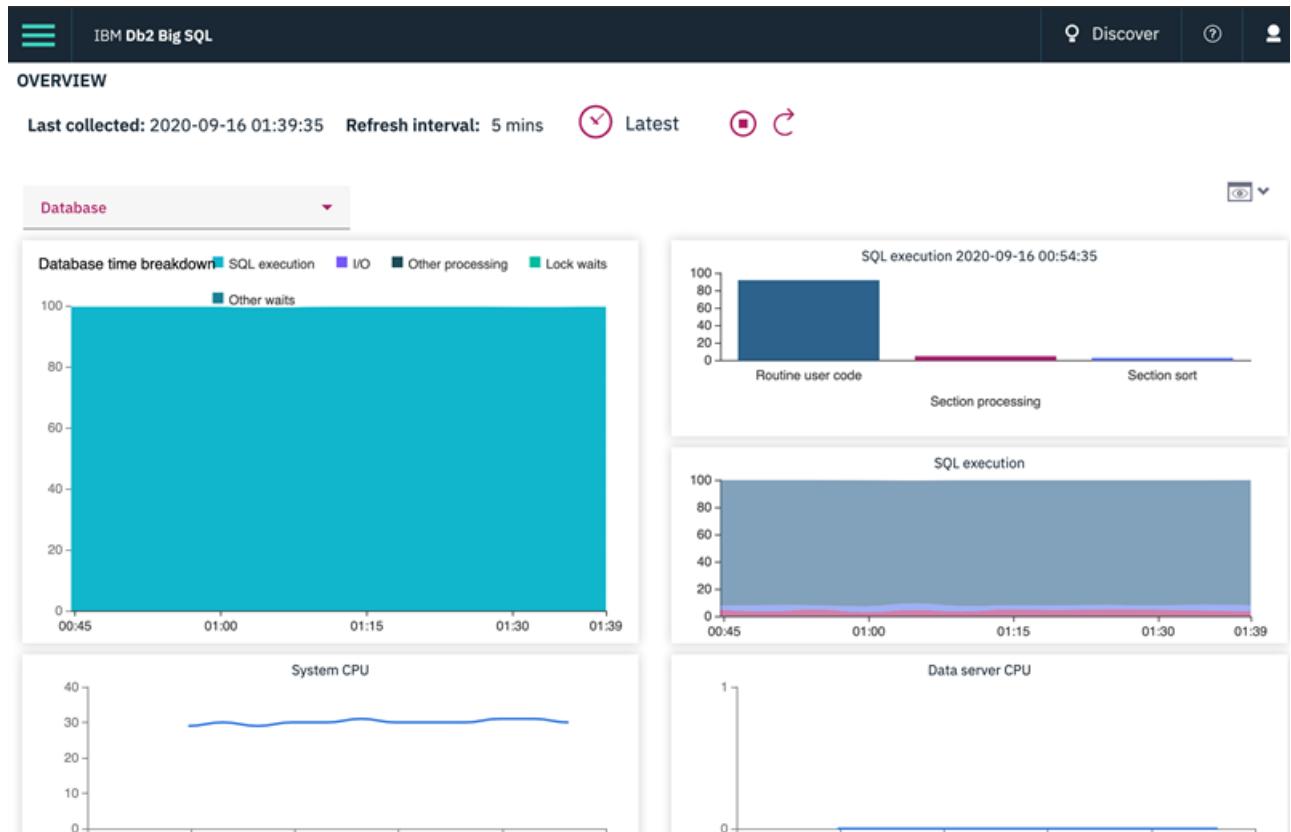
**RUN SQL**

**MONITOR**

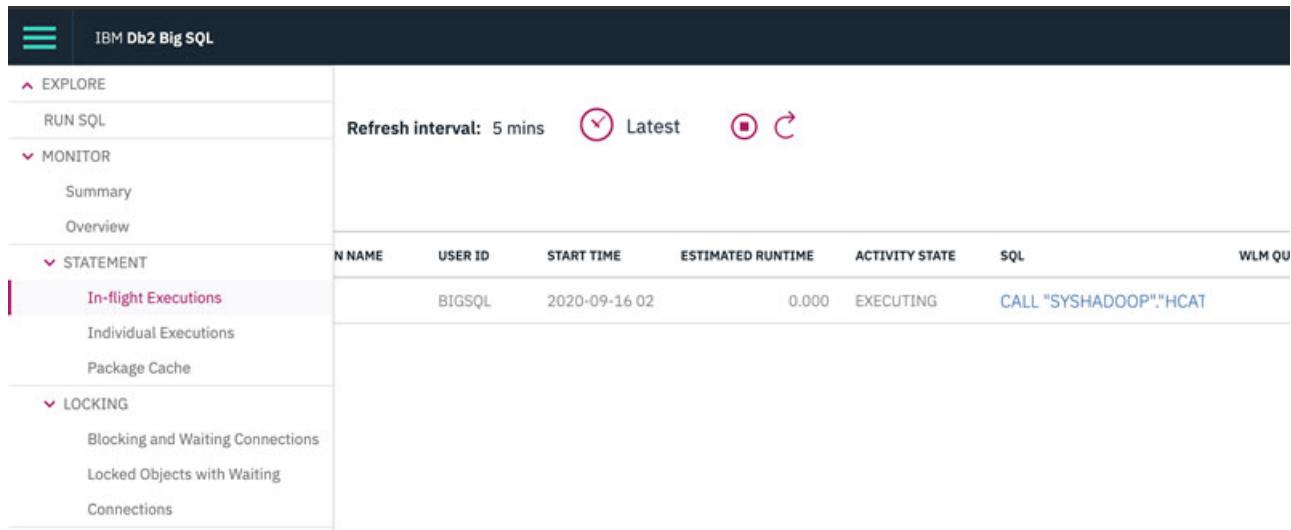
Summary

**Overview**

The Overview page is displayed.



- 7. Explore the other monitoring capabilities. Click **MONITOR -> STATEMENT -> In-flight Executions**. Then, continue to explore the other monitoring options in the menu.



- 8. From the menu at the upper left, click **Run SQL**.

The screenshot shows the IBM Db2 Big SQL interface. On the left is a navigation sidebar with options: EXPLORE, RUN SQL (which is selected), MONITOR, SETTINGS, CONNECTION INFO, and HELP. The main area has a toolbar with icons for file operations and syntax assistance. A status bar at the bottom right shows "Result - 09/16/20 01:30:28". The results pane displays a single row from a query: "SELECT count(\*)". The row contains two columns: "Result set 1" with values "1" and "781".

\_\_\_ 9. Select **Blank** to open an empty SQL window.

Add new script

[Choose the way to create](#) [Open a script to edit](#)

[+ Blank](#)

[↑ From file](#)

Templates  
Choose one template to start your SQL editor.

\_\_\_ 10. Copy the following statements in the RUN SQL window.

```
select * from dual;
select tabschema, tablename from syscat.tables fetch first 5 rows only;
```

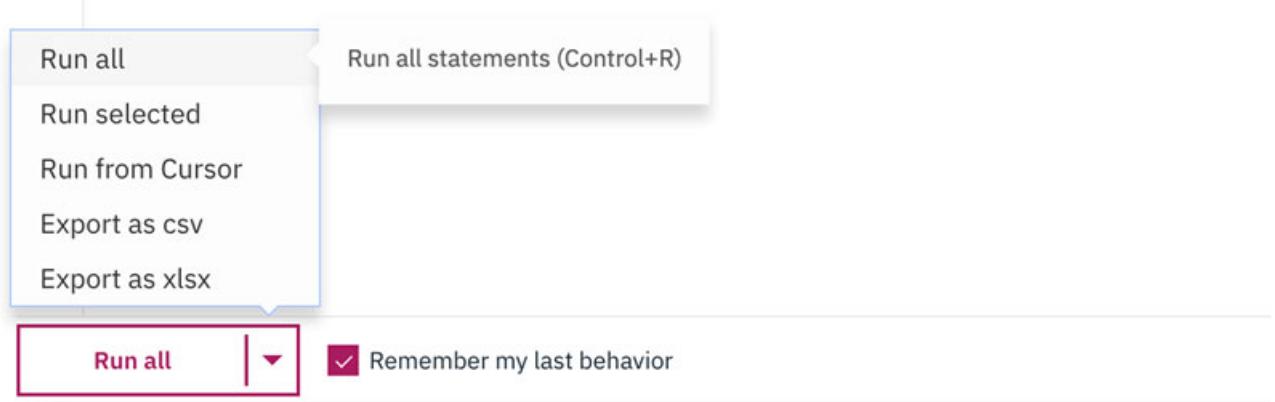
## RUN SQL

\* Untitled - 1



```
1 select * from dual;
2 select tabschema, tablename from syscat.tables fetch first 5 rows only;
3
```

\_\_\_ 11. Select **Run All** from the **Run** drop down at the lower left of the page.



12. Analyze the results that are returned to the browser. For example, observe the columns, number of records, run time and any other result that is returned.

The screenshot shows the IBM Db2 Big SQL interface with two query results displayed. The first query is:

```
1 select * from dual;
2 select tabschema, tablename from syscat.tables fetch first 5 row...
```

The result for the first query is:

Result set 1	
DUMMY	X

The result for the second query is:

TABSCHEMA	TABNAME
BIGSQL	SMOKE_HADOOP2_1525422353
BIGSQL	SMOKE_HADOOP2_1639502892
DB_STUDENT0000	CHICAGO_BUS
IBMCORSOLE	ALERT
IBMCORSOLE	ALERT_PROPERTIES

## End of exercise

## Exercise review and wrap-up

In this exercise, you became familiar with the Ambari Web UI and checked services status.

You used the JSqsh shell and set up the connection information for Db2 Big SQL.

You became familiar with the JSqsh shell and its documentation and command syntax. You ran Db2 Big SQL statements from the JSqsh shell.

Finally, you started the Db2 Big SQL console from the Ambari Web UI and explored several monitoring options and ran SQL statements.

# Exercise 2. Creating and managing Db2 Big SQL schemas and tables

## Estimated time

01:30

## Overview

In this exercise, you start by creating and dropping a simple Hadoop table by using Db2 Big SQL. Then, you create multiple tables by using various data types and load the tables with data. You also work with views, external tables, and other methods of creating Db2 Big SQL tables.

## Objectives

After completing this exercise, you will be able to:

- Create and drop a simple Hadoop table.
- Work with Db2 Big SQL sample data and create sample tables.
- Copy sample data into DHFS.
- Load sample data into tables by using Db2 Big SQL.
- Create views that span multiple tables.
- Query views.
- Populate a table with data based on the results of a query by using Db2 Big SQL.
- Create and work with external tables.

## Introduction

Db2 Big SQL is an SQL language processor to summarize, query, and analyze data in an Apache Hadoop distributed file system. These queries are low-latency queries that return information quickly to reduce response time and provide improved access to data.

You can run the SQL scripts that are included in the Db2 Big SQL installation from JSqsh or Db2 Big SQL Console to create tables and load data. In this exercise, you explore some of the basic Db2 Big SQL queries and other Db2 Big SQL capabilities to create, manage, and load tables.

## Requirements

- PuTTY SSH client installed in your workstation.
- For information about the variables used in this exercise, refer to the table in Exercise 1. “Connecting to the IBM Db2 Big SQL server”.

## Exercise instructions

In this exercise, you complete the following tasks:

- \_\_\_ 1. Create and drop a simple Db2 Big SQL table.
- \_\_\_ 2. Create sample tables.
- \_\_\_ 3. Move data into HDFS.
- \_\_\_ 4. Load data into Db2 Big SQL tables.
- \_\_\_ 5. Create and work with views.
- \_\_\_ 6. Populate a table with 'INSERT INTO ... SELECT'.
- \_\_\_ 7. Work with external tables.

### **Part 1: Creating and dropping a simple Db2 Big SQL table**

In this part, you use the JSqsh shell to perform various operations such as connecting to the **bigsq1** database, creating a simple Hadoop table, inserting rows, and dropping the table.

Complete the following steps:

- \_\_\_ 1. Open **PuTTY** to connect to the VM. Enter the <hostname>, <username>, and <password>.
- \_\_\_ 2. Start the JSqsh shell.  

```
/usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh
```
- \_\_\_ 3. From the **JSqsh** shell, connect to the Db2 Big SQL server by using the connection **bigsq1** that you created in a previous exercise.  

```
\connect bigsql
```

If prompted for a password, enter your password <password> .
- \_\_\_ 4. Run the following command in JSqsh to create a simple Hadoop table:  

```
create hadoop table test1 (col1 int, col2 varchar(5));
```

The results are similar to the following output:

```
0 rows affected (total: 1.026s)
```

Because you didn't specify a schema name for the table, it was created in your default schema, which is the user <username> specified in your JDBC connection. This statement is equivalent to

```
create hadoop table yourID.test1 (col1 int, col2 varchar(5));
```

Where **yourID** is the user name <username> for your connection.

- \_\_\_ 5. Display all tables in the current schema with the **\tables** command.  

```
\tables
```

```
[dataengineer.ibm.com] [student0001] 1> \tables
+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | TABLE_TYPE |
+-----+-----+-----+
| STUDENT0001 | TEST1      | TABLE      |
+-----+-----+-----+
```

\_\_\_ 6. Insert a row into your table.

```
insert into test1 values (1, 'one');
```

The results are similar to the following output:

```
1 row affected (total: 5.224s)
```

Use this form of the INSERT statement (INSERT INTO...VALUES...) only for test purposes because the operation is not parallelized on your cluster. To populate a table with data in a manner that leverages parallel processing, use the following commands:

- Db2 Big SQL LOAD command.
- INSERT INTO...SELECT FROM statement.
- CREATE TABLE AS...SELECT statement.

You learn more about these commands later.

\_\_\_ 7. To view the metadata about a table, use the \describe command with the table name.

```
\describe TEST1
```

The results are similar to the following output:

```
[dataengineer.ibm.com] [student0001] 1> \describe TEST1
+-----+-----+-----+-----+-----+-----+
| TABLE_SCHEMA | COLUMN_NAME | TYPE_NAME | COLUMN_SIZE | DECIMAL_DIGITS | IS_NULLABLE |
+-----+-----+-----+-----+-----+-----+
| STUDENT0001 | COL1        | INTEGER    |          10 |             0 | YES         |
| STUDENT0001 | COL2        | VARCHAR    |           5 | [NULL]       | YES         |
+-----+-----+-----+-----+-----+-----+
```

\_\_\_ 8. Optionally, query the system for metadata about this table:

```
select tabschema, colname, colno, typename, length
from syscat.columns
where tabschema = USER and tablename= 'TEST1';
```

You can split the query across multiple lines in the JSqsh shell. When you press **Enter**, the shell provides another line for you to continue your command or SQL statement. A semi-colon or **go** command causes your SQL statement to run.

The results are similar to the following output:

```
[dataengineer.ibm.com] [student0001] 1> select tabschema, colname, colno, typename, length
[dataengineer.ibm.com] [student0001] 2> from syscat.columns
[dataengineer.ibm.com] [student0001] 3> where tabschema = USER and tablename= 'TEST1';
+-----+-----+-----+-----+
| TABSCHEMA | COLNAME | COLNO | TYPENAME | LENGTH |
+-----+-----+-----+-----+
| STUDENT0001 | COL1 | 0 | INTEGER | 4 |
| STUDENT0001 | COL2 | 1 | VARCHAR | 5 |
+-----+-----+-----+-----+
2 rows in results(first row: 0.006s; total: 0.008s)
```

SYSCAT.COLUMNS is one of a number of views that are supplied over system catalog tables, which is automatically maintained for you by the Db2 Big SQL service.

- \_\_\_ 9. Issue the following command to drop the table:

```
drop table test1;
```

The results are similar to the following output:

```
0 rows affected (total: 0.514s)
```

- \_\_\_ 10. Run \tables to ensure that the table was in fact dropped.

```
[eddev27.canlab.ibm.com][bigsq1] 1> drop table test1;
0 rows affected (total: 0.925s)
[eddev27.canlab.ibm.com][bigsq1] 1> \tables
+-----+-----+-----+
| TABLE_SCHEM | TABLE_NAME | TABLE_TYPE |
+-----+-----+-----+
+-----+-----+-----+
```

## **Part 2: Creating sample tables**

---



### **Attention**

If students encounter the following error when they try to create a table: “**The transaction log for the database is full.. SQLCODE=-9964, SQLSTATE=57011.**” the instructor must complete the following steps by logging in with the **bigsq1** user and <**bigsq1 password**>

1. Login as **bigsq1** user to the Db2 Big SQL head node.

2. Run the following commands

```
db2 connect to bigsq1
db2 get db cfg for bigsq1 | grep LOGFILSZ (make a note of the value)
db2 update db cfg for bigsq1 using LOGFILSZ <2* the previous value>
```

3. Restart Db2 Big SQL

- a. You can restart the IBM Db2 Big SQL service from the Ambari Web UI by logging in as the <**Ambari admin**> user.

- b. Alternatively, restart Db2 Big SQL by running the following commands:

```
db2stop force  
db2start
```

---

In this part, you create several sample tables.

Determine the location of the Db2 Big SQL sample data in your server's local Linux file system and make a note of it. You need to use this path specification when you issue LOAD commands later in this exercise.

Subsequent examples in this section presume your sample data is in the directory `/usr/ibmpacks/bigrsql/6.0.0.0/bigrsql/samples/data`. This is the location of the data in typical Db2 Big SQL installations.

Also, note that the statements in this exercise are provided in the scripts that are located in `/home/bigrsql/labfiles/bigrsql/BIG_SQL_Data_Analysis`. You can copy and paste the statements into your console of choice. The following CREATE statements can be found in the file `1_CREATE_TABLE_statements.sql`.

- 1. Using your choice of either JSqsh or Big SQL Console web interface, establish a connection to your Db2 Big SQL server. Refer to “Exercise 1. Part 3. Connecting to the Db2 Big SQL server by using JSqsh” and “Exercise 1. Part 6. Exploring Db2 Big SQL through Ambari by using the Db2 Big SQL console”
- 2. Create *eight* tables in your default schema. Issue each of the following `CREATE TABLE` statements one at a time, and verify that each statement completes successfully.

- Dimension table for region information

```

CREATE HADOOP TABLE IF NOT EXISTS go_region_dim
(
    country_key      INT NOT NULL
    , country_code   INT NOT NULL
    , flag_image     VARCHAR(45)
    , iso_three_letter_code VARCHAR(9) NOT NULL
    , iso_two_letter_code  VARCHAR(6) NOT NULL
    , iso_three_digit_code VARCHAR(9) NOT NULL
    , region_key     INT NOT NULL
    , region_code    INT NOT NULL
    , region_en      VARCHAR(90) NOT NULL
    , country_en     VARCHAR(90) NOT NULL
    , region_de      VARCHAR(90), country_de VARCHAR(90), region_fr      VARCHAR(90)
    , country_fr     VARCHAR(90), region_ja  VARCHAR(90), country_ja  VARCHAR(90)
    , region_cs      VARCHAR(90), country_cs VARCHAR(90), region_da      VARCHAR(90)
    , country_da     VARCHAR(90), region_el  VARCHAR(90), country_el  VARCHAR(90)
    , region_es      VARCHAR(90), country_es VARCHAR(90), region_fi      VARCHAR(90)
    , country_fi     VARCHAR(90), region_hu  VARCHAR(90), country_hu  VARCHAR(90)
    , region_id      VARCHAR(90), country_id VARCHAR(90), region_it      VARCHAR(90)
    , country_it     VARCHAR(90), region_ko  VARCHAR(90), country_ko  VARCHAR(90)
    , region_ms      VARCHAR(90), country_ms VARCHAR(90), region_nl      VARCHAR(90)
    , country_nl     VARCHAR(90), region_no  VARCHAR(90), country_no  VARCHAR(90)
    , region_pl      VARCHAR(90), country_pl VARCHAR(90), region_pt      VARCHAR(90)
    , country_pt     VARCHAR(90), region_ru  VARCHAR(90), country_ru  VARCHAR(90)
    , region_sc      VARCHAR(90), country_sc VARCHAR(90), region_sv      VARCHAR(90)
    , country_sv     VARCHAR(90), region_tc  VARCHAR(90), country_tc  VARCHAR(90)
    , region_th      VARCHAR(90), country_th VARCHAR(90)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;

```

- Dimension table tracking method of order for the sale, for example, Web or fax

```

CREATE HADOOP TABLE IF NOT EXISTS sls_order_method_dim
( order_method_key      INT NOT NULL
, order_method_code     INT NOT NULL
, order_method_en VARCHAR(90) NOT NULL
, order_method_de VARCHAR(90), order_method_fr  VARCHAR(90)
, order_method_ja VARCHAR(90), order_method_cs  VARCHAR(90)
, order_method_da VARCHAR(90), order_method_el  VARCHAR(90)
, order_method_es VARCHAR(90), order_method_fi  VARCHAR(90)
, order_method_hu VARCHAR(90), order_method_id  VARCHAR(90)
, order_method_it VARCHAR(90), order_method_ko  VARCHAR(90)
, order_method_ms VARCHAR(90), order_method_nl  VARCHAR(90)
, order_method_no VARCHAR(90), order_method_pl  VARCHAR(90)
, order_method_pt VARCHAR(90), order_method_ru  VARCHAR(90)
, order_method_sc VARCHAR(90), order_method_sv  VARCHAR(90)
, order_method_tc VARCHAR(90), order_method_th  VARCHAR(90)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;

```

- Look up table with product brand information in various languages

```

CREATE HADOOP TABLE IF NOT EXISTS sls_product_brand_lookup
( product_brand_code    INT NOT NULL
, product_brand_en      VARCHAR(90) NOT NULL
, product_brand_de      VARCHAR(90), product_brand_fr  VARCHAR(90)
, product_brand_ja      VARCHAR(90), product_brand_cs  VARCHAR(90)
, product_brand_da      VARCHAR(90), product_brand_el  VARCHAR(90)
, product_brand_es      VARCHAR(90), product_brand_fi  VARCHAR(90)
, product_brand_hu      VARCHAR(90), product_brand_id  VARCHAR(90)
, product_brand_it      VARCHAR(90), product_brand_ko  VARCHAR(90)
, product_brand_ms      VARCHAR(90), product_brand_nl  VARCHAR(90)
, product_brand_no      VARCHAR(90), product_brand_pl  VARCHAR(90)
, product_brand_pt      VARCHAR(90), product_brand_ru  VARCHAR(90)
, product_brand_sc      VARCHAR(90), product_brand_sv  VARCHAR(90)
, product_brand_tc      VARCHAR(90), product_brand_th  VARCHAR(90)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;

```

- Product dimension table

```

CREATE HADOOP TABLE IF NOT EXISTS sls_product_dim
( product_key          INT NOT NULL
, product_line_code    INT NOT NULL
, product_type_key    INT NOT NULL
, product_type_code    INT NOT NULL
, product_number      INT NOT NULL
, base_product_key    INT NOT NULL
, base_product_number INT NOT NULL
, product_color_code  INT
, product_size_code   INT
, product_brand_key   INT NOT NULL
, product_brand_code  INT NOT NULL
, product_image        VARCHAR(60)
, introduction_date   TIMESTAMP
, discontinued_date   TIMESTAMP
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;

```

- Look up table with product line info in various languages

```

CREATE HADOOP TABLE IF NOT EXISTS sls_product_line_lookup
( product_line_code    INT NOT NULL
, product_line_en       VARCHAR(90) NOT NULL
, product_line_de       VARCHAR(90), product_line_fr  VARCHAR(90)
, product_line_ja       VARCHAR(90), product_line_cs  VARCHAR(90)
, product_line_da       VARCHAR(90), product_line_el  VARCHAR(90)
, product_line_es       VARCHAR(90), product_line_fi  VARCHAR(90)
, product_line_hu       VARCHAR(90), product_line_id  VARCHAR(90)
, product_line_it       VARCHAR(90), product_line_ko  VARCHAR(90)
, product_line_ms       VARCHAR(90), product_line_nl  VARCHAR(90)
, product_line_no       VARCHAR(90), product_line_pl  VARCHAR(90)
, product_line_pt       VARCHAR(90), product_line_ru  VARCHAR(90)
, product_line_sc       VARCHAR(90), product_line_sv  VARCHAR(90)
, product_line_tc       VARCHAR(90), product_line_th  VARCHAR(90)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

```

- Look up table for products

```
CREATE HADOOP TABLE IF NOT EXISTS sls_product_lookup
( product_number      INT NOT NULL
, product_language    VARCHAR(30) NOT NULL
, product_name        VARCHAR(150) NOT NULL
, product_description VARCHAR(765)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

- Fact table for sales

```
CREATE HADOOP TABLE IF NOT EXISTS sls_sales_fact
( order_day_key      INT NOT NULL
, organization_key    INT NOT NULL
, employee_key        INT NOT NULL
, retailer_key         INT NOT NULL
, retailer_site_key   INT NOT NULL
, product_key          INT NOT NULL
, promotion_key        INT NOT NULL
, order_method_key    INT NOT NULL
, sales_order_key     INT NOT NULL
, ship_day_key        INT NOT NULL
, close_day_key       INT NOT NULL
, quantity             INT
, unit_cost            DOUBLE
, unit_price           DOUBLE
, unit_sale_price      DOUBLE
, gross_margin          DOUBLE
, sale_total            DOUBLE
, gross_profit          DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
; 
```

- Fact table for marketing promotions

```
CREATE HADOOP TABLE IF NOT EXISTS mrk_promotion_fact
( organization_key      INT NOT NULL
, order_day_key        INT NOT NULL
, rtl_country_key      INT NOT NULL
, employee_key         INT NOT NULL
, retailer_key          INT NOT NULL
, product_key           INT NOT NULL
, promotion_key         INT NOT NULL
, sales_order_key       INT NOT NULL
, quantity              SMALLINT
, unit_cost              DOUBLE
, unit_price             DOUBLE
, unit_sale_price        DOUBLE
, gross_margin            DOUBLE
, sale_total              DOUBLE
, gross_profit             DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

The following figure shows the CREATE HADOOP TABLE statement, which defines a Db2 Big SQL table that is based on a Hive table for the Hadoop environment. This statement is entered in the Db2 Big SQL console and run successfully.

The screenshot shows the IBM Db2 Big SQL interface. At the top, there's a navigation bar with icons for Home, Recent, and Help. Below it is a toolbar with various icons like Run, Stop, Refresh, and Save. The main area is titled "RUN SQL" and contains a code editor with the following SQL script:

```

1 CREATE HADOOP TABLE IF NOT EXISTS go_region_dim
2 ( country_key INT NOT NULL
3 , country_code INT NOT NULL
4 , flag_image VARCHAR(45)
5 , iso_three_letter_code VARCHAR(9) NOT NULL
6 , iso_two_letter_code VARCHAR(6) NOT NULL
7 , iso_three_digit_code VARCHAR(9) NOT NULL
8 , region_key INT NOT NULL
9 , region_code INT NOT NULL
10 , region_en VARCHAR(90) NOT NULL
11 , country_en VARCHAR(90) NOT NULL
12 , region_de VARCHAR(90), country_de VARCHAR(90), region_fr VARCHAR(90)
13 , country_fr VARCHAR(90), region_ja VARCHAR(90), country_ja VARCHAR(90)
14 , region_cs VARCHAR(90), country_cs VARCHAR(90), region_da VARCHAR(90)
15 , country_da VARCHAR(90), region_el VARCHAR(90), country_el VARCHAR(90)
16 , region_es VARCHAR(90), country_es VARCHAR(90), region_fi VARCHAR(90)
17 , country_fi VARCHAR(90), region_hu VARCHAR(90), country_hu VARCHAR(90)
18 , region_id VARCHAR(90), country_id VARCHAR(90), region_it VARCHAR(90)
19 , country_it VARCHAR(90), region_ko VARCHAR(90), country_ko VARCHAR(90)
20 , region_ms VARCHAR(90), country_ms VARCHAR(90), region_nl VARCHAR(90)
21 , country_nl VARCHAR(90), region_no VARCHAR(90), country_no VARCHAR(90)
22 , region_pl VARCHAR(90), country_pl VARCHAR(90), region_pt VARCHAR(90)
23 , country_pt VARCHAR(90), region_ru VARCHAR(90), country_ru VARCHAR(90)
24 , region_sc VARCHAR(90), country_sc VARCHAR(90), region_sv VARCHAR(90)
25 , country_sv VARCHAR(90), region_tc VARCHAR(90), country_tc VARCHAR(90)
26 , region_th VARCHAR(90), country_th VARCHAR(90)
27 )
28 ROW FORMAT DELIMITED
29 FIELDS TERMINATED BY '\t'
30 LINES TERMINATED BY '\n'
31 STORED AS TEXTFILE
32 ;
33

```

Below the code editor is a toolbar with icons for Discover, Help, and User. The main content area has tabs for "Script Library" and "Result History". Under "Result History", there's a result set for "Result - 09/16/2018:18:18:10". It shows the executed SQL statement and its success status:

CREATE HADOOP TABLE IF NOT EXISTS go\_region\_dim ( country\_key INT NOT N...  
Run time: 0.092s  
Status: Success | Affected Rows: 0

Let's briefly explore some aspects of the CREATE TABLE statements that are shown here. If you have SQL background, most of these statements are familiar to you. However, after the column specification, there are some additional clauses that are unique to Db2 Big SQL. They are clauses that enable it to exploit Hadoop storage mechanisms, in this case, Hive. The ROW FORMAT clause specifies that fields are to be terminated by tabs ("\\t") and lines are to be

terminated by new line characters ("\\n"). The table is stored in a TEXTFILE format, making it easy for a wide range of applications to work with it.

- 3. If you are within JSqsh, run the \tables command to see a listing of all the new tables.

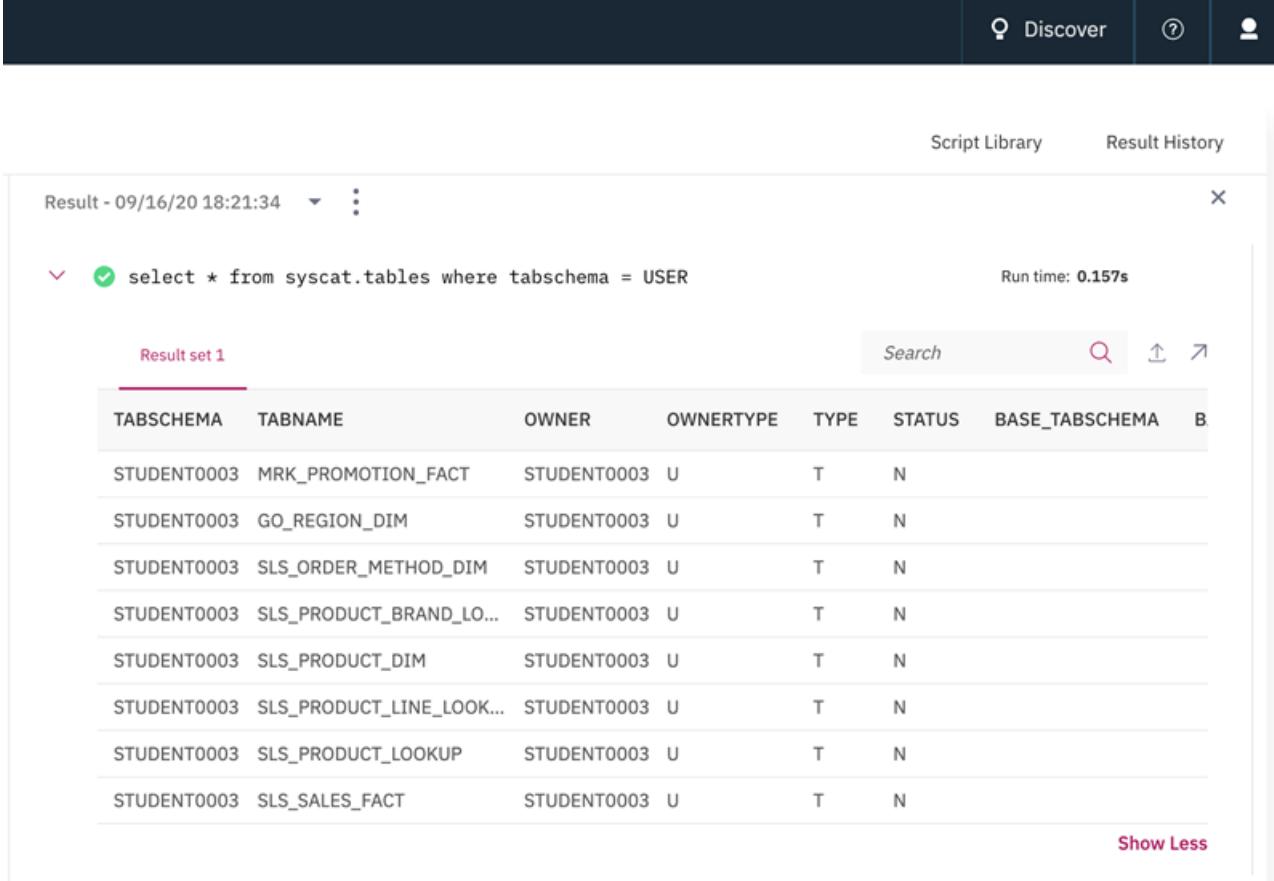
If you are using the Db2 Big SQL Console, run the following query:

```
select * from syscat.tables where tabschema = USER;
```



### Note

Click the **View More** icon  to display all the tables.



The screenshot shows the Db2 Big SQL console interface. At the top, there are navigation links: Discover, Help, and User. Below that, the title bar says "Result - 09/16/20 18:21:34". On the right, there are buttons for Script Library and Result History. The main area displays the query "select \* from syscat.tables where tabschema = USER" and its execution time "Run time: 0.157s". The result set is titled "Result set 1" and contains the following data:

TABSCHEMA	TABNAME	OWNER	OWNERTYPE	TYPE	STATUS	BASE_TABSCHEMA	B
STUDENT0003	MRK_PROMOTION_FACT	STUDENT0003	U	T	N		
STUDENT0003	GO_REGION_DIM	STUDENT0003	U	T	N		
STUDENT0003	SLS_ORDER_METHOD_DIM	STUDENT0003	U	T	N		
STUDENT0003	SLS_PRODUCT_BRAND_LO...	STUDENT0003	U	T	N		
STUDENT0003	SLS_PRODUCT_DIM	STUDENT0003	U	T	N		
STUDENT0003	SLS_PRODUCT_LINE_LOOK...	STUDENT0003	U	T	N		
STUDENT0003	SLS_PRODUCT_LOOKUP	STUDENT0003	U	T	N		
STUDENT0003	SLS_SALES_FACT	STUDENT0003	U	T	N		

At the bottom right of the result set, there is a "Show Less" link.

## Part 3: Moving data into HDFS

You can find plenty of sample data in the folder

`/usr/ibmpacks/bigrsql/6.0.0.0/bigrsql/samples/data/`.

In this part, you copy sample data into the HDFS, and later load it into tables by using Db2 Big SQL.

- 1. Go back to the PuTTY session and quit the JSqsh shell by typing `quit`.

**Note**

If your PuTTY session is expired, restart it.

- 
- \_\_\_ 2. Create a directory on HDFS to hold the sample data files by running the following command:

```
hadoop fs -mkdir /user/<username>/sampledata
```

- \_\_\_ 3. Run the following command to ensure that the new directory is successfully created.

```
hadoop fs -ls /user/<username>/
```

The results are similar to the following output.

```
[student0002@dataengineer ~]$ hadoop fs -ls /user/student0002/
Found 7 items
drwx-----  - student0002 hdfs          0 2020-10-18 01:00 /user/student0002/.Trash
drwxr-xr-x  - student0002 hdfs          0 2020-10-17 15:53 /user/student0002/.sparkStaging
drwx-----  - student0002 hdfs          0 2020-10-17 15:12 /user/student0002/.staging
drwxr-xr-x  - student0002 hdfs          0 2020-10-16 16:04 /user/student0002/Gutenberg
drwxrwxrwx  - student0002 hdfs          0 2020-10-19 03:06 /user/student0002/bigsql_lab
-rw-r--r--  3 student0002 hdfs         19 2020-10-17 15:07 /user/student0002/patternsToSkip
drwxr-xr-x  - student0002 hdfs          0 2020-10-18 23:43 /user/student0002/sampledata
```

- \_\_\_ 4. Load the data into HDFS.

```
hadoop fs -copyFromLocal /usr/ibmpacks/bigsql/6.0.0.0/bigsql/samples/data/
/user/<username>/sampledata
```

- \_\_\_ 5. List the contents of the newly created data directory on HDFS.

```
hadoop fs -ls /user/<username>/sampledata/data
```

Observe that dozens of data files that were copied. The following figure shows partial results.

```
[student0002@dataengineer ~]$ hadoop fs -ls /user/student0002/sampleddata/data
Found 70 items
-rw-r--r--  3 student0002 hdfs      821 2020-10-18 23:43 /user/student0002/sampleddata/data/Disclaimer.txt
-rw-r--r--  3 student0002 hdfs      64  2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.BURST_TABLE.txt
-rw-r--r--  3 student0002 hdfs     228 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.BURST_TABLE2.txt
-rw-r--r--  3 student0002 hdfs  2841950 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.DIST_INVENTORY_FACT.txt
-rw-r--r--  3 student0002 hdfs 4783840 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.DIST_PRODUCT_FORECAST_FACT.txt
-rw-r--r--  3 student0002 hdfs 564113 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.DIST_RETURNED_ITEMS_FACT.txt
-rw-r--r--  3 student0002 hdfs 2796 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.DIST_RETURN_REASON_DIM.txt
-rw-r--r--  3 student0002 hdfs 535639 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_EMPLOYEE_DIM.txt
-rw-r--r--  3 student0002 hdfs 6211856 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_EXPENSE_FACT.txt
-rw-r--r--  3 student0002 hdfs 995660 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_EXPENSE_PLAN_FACT.txt
-rw-r--r--  3 student0002 hdfs 27895 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_EXPENSE_TYPE_DIM.txt
-rw-r--r--  3 student0002 hdfs 789 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_EXPENSE_UNIT_LOOKUP.txt
-rw-r--r--  3 student0002 hdfs 2719 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_POSITION_DIM.txt
-rw-r--r--  3 student0002 hdfs 32857 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_POSITION_LOOKUP.txt
-rw-r--r--  3 student0002 hdfs 466818 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_POSITION_SUMMARY_FACT.txt
-rw-r--r--  3 student0002 hdfs 1285 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_RANKING_DIM.txt
-rw-r--r--  3 student0002 hdfs 62601 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_RANKING_FACT.txt
-rw-r--r--  3 student0002 hdfs 12669 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_RECRUITMENT_DIM.txt
-rw-r--r--  3 student0002 hdfs 40427 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_RECRUITMENT_FACT.txt
-rw-r--r--  3 student0002 hdfs 9250 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SUCCESSION_FACT.txt
-rw-r--r--  3 student0002 hdfs 1785 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SUCCESSION_STATUS_DIM.txt
-rw-r--r--  3 student0002 hdfs 1245877 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SUMMARY_FACT.txt
-rw-r--r--  3 student0002 hdfs 205453 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SURVEY_FACT.txt
-rw-r--r--  3 student0002 hdfs 461 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SURVEY_TARG_FACT.txt
-rw-r--r--  3 student0002 hdfs 2576 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_SURVEY_TOPIC_DIM.txt
-rw-r--r--  3 student0002 hdfs 2360 2020-10-18 23:43 /user/student0002/sampleddata/data/GOSALESOW.EMP_TERMINATION_LOOKUP.txt
```

## Part 4: Loading data into Db2 Big SQL tables

In this part, you load data into each of following tables by using the sample data provided. Note that LOAD returns a warning message that provides details on the number of rows loaded.

You can use either the JSqsh shell or the Db2 Big SQL console.



### Important

**LONG RUNNING commands.** This part of the exercise includes running multiple LOAD commands. When a single user submits a LOAD command, it usually completes in about 1 minute. In the current class environment, when tens of students run commands at the same time, more wait time is to be expected. In our tests, the maximum wait time experienced was approximate 7 minutes. Be patient and do not close or break your session.

You can copy and paste the following statement into your JSqsh session or Db2 Big SQL Console window.



### Note

You can also obtain the code from the file `2_Load_Statements.sql`, which is located in the folder `/home/bigrsql/labfiles/bigrsql/BIG_SQL_Data_Analysis`.

If you use the file, change `/user/bigrsql` to `/user/<username>` before running the commands.

- \_\_ 1. Open the Db2 Big SQL Console as you did in Exercise 1 Part 6.

- 2. Issue the first LOAD statement and verify that the operation completes successfully.

```
load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.GO_REGION_DIM.txt' with SOURCE
PROPERTIES ('field.delimiter'='\t') INTO TABLE GO_REGION_DIM overwrite;
```

Don't be alarmed if you observe a warning message that displays information about the loading of data to the Hadoop table. Observe that 21 rows were processed. The following figure shows the message on Data Server Manager (DSM):

The screenshot shows the Data Server Manager (DSM) interface. At the top, there are navigation links: Discover, Help, and User. Below the header, there are tabs for Script Library and Result History. The main area displays a result set from a query. The title bar says "Result - 09/16/20 19:47:54". The results show a single row:

▼	⚠ load hadoop using file url '/user/student0003/sampledata/data/GOSALESdw...' Run time: 50.299s
Status: <b>Warning</b>   Affected Rows: 0 <b>Warning message</b> Loading of data to a Hadoop table or processing of data in an external table completed. Number of rows processed: "21". Number of source records: "21". If the source was a file, number of skipped lines: "0". Number of rejected source records: "0". Job or file identifier: "job_1599558744707_0003". SQLCODE=5108, SQLSTATE=0 , DRIVER=4.24.92 <a href="#">Learn more about this warning</a>	

The command loads data into a table by using data from HDFS. If you omit the `hdfs://` URI as in this example, the LOAD HADOOP USING command assumes that it is an HDFS scheme.

The WITH SOURCE PROPERTIES clause specifies that fields in the source data are delimited by tabs ("\"t"). The INTO TABLE clause identifies the target table for the LOAD operation.

The OVERWRITE keyword indicates that any existing data in the table will be replaced by data from the source file. If you want to add rows to the table, you can specify APPEND instead.

- 3. Issue each of the following LOAD statements, one at a time, and verify that the operation completes successfully.

```
load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.SLS_ORDER_METHOD_DIM.txt' with SOURCE
PROPERTIES ('field.delimiter'='\t') INTO TABLE SLS_ORDER_METHOD_DIM
overwrite;
```

```
load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.SLS_PRODUCT_BRAND_LOOKUP.txt'
```

```
with SOURCE PROPERTIES ('field.delimiter'='\t') INTO TABLE  
SLS_PRODUCT_BRAND_LOOKUP overwrite;  
  
load hadoop using file url  
'/user/<username>/sampledata/data/GOSALESdw.SLS_PRODUCT_DIM.txt' with SOURCE  
PROPERTIES ('field.delimiter'='\t') INTO TABLE SLS_PRODUCT_DIM overwrite;
```

```

load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.SLS_PRODUCT_LINE_LOOKUP.txt'
with SOURCE PROPERTIES ('field.delimiter'='\t') INTO TABLE
SLS_PRODUCT_LINE_LOOKUP overwrite;

load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.SLS_PRODUCT_LOOKUP.txt' with
SOURCE PROPERTIES ('field.delimiter'='\t') INTO TABLE SLS_PRODUCT_LOOKUP
overwrite;

load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.SLS_SALES_FACT.txt' with SOURCE
PROPERTIES ('field.delimiter'='\t') INTO TABLE SLS_SALES_FACT overwrite;

load hadoop using file url
'/user/<username>/sampledata/data/GOSALESdw.MRK_PROMOTION_FACT.txt' with
SOURCE PROPERTIES ('field.delimiter'='\t') INTO TABLE MRK_PROMOTION_FACT
overwrite;

```

- 4. Query each table to verify that the expected number of rows was loaded into each table. Run each query that follows individually and compare the results with the number of rows that are specified in the comment line that precedes each query. The file **3\_SELECT\_statements - validating LOAD.sql** is located in the folder **/home/bigsql/labfiles/bigsql/BIG\_SQL\_Data\_Analysis**.

```

-- total rows in GO_REGION_DIM = 21
select count(*) from GO_REGION_DIM;

-- total rows in sls_order_method_dim = 7
select count(*) from sls_order_method_dim;

-- total rows in SLS_PRODUCT_BRAND_LOOKUP = 28
select count(*) from SLS_PRODUCT_BRAND_LOOKUP;

-- total rows in SLS_PRODUCT_DIM = 274
select count(*) from SLS_PRODUCT_DIM;

-- total rows in SLS_PRODUCT_LINE_LOOKUP = 5
select count(*) from SLS_PRODUCT_LINE_LOOKUP;

-- total rows in SLS_PRODUCT_LOOKUP = 6302
select count(*) from SLS_PRODUCT_LOOKUP;

-- total rows in SLS_SALES_FACT = 446023
select count(*) from SLS_SALES_FACT;

-- total rows gosalesdw.MRK_PROMOTION_FACT = 11034
select count(*) from MRK_PROMOTION_FACT;

```

## Part 5: Creating and working with views

Db2 Big SQL supports views (virtual tables) based on one or more physical tables. In this part, you create a view that spans multiple tables. Then, you query this view by using a simple SELECT statement. By doing so, you discover that you can work with views in Db2 Big SQL much as you can work with views in a relational DBMS.

The file `5_VIEW_statements.sql` is located in the folder `/home/bigsq1/labfiles/bigsq1/BIG_SQL_Data_Analysis`.

- 1. Use either the Db2 Big SQL Console or JSqsh to create a view that is named MYVIEW that extracts information about product sales that are featured in marketing promotions. Since the schema name is omitted in both the CREATE and FROM object names, the current schema (your user name), is assumed.

```
create view myview as
  select product_name, sales.product_key, mkt.quantity,
         sales.order_day_key, sales.sales_order_key, order_method_en
    from
      mrk_promotion_fact mkt,
           sls_sales_fact sales,
           sls_product_dim prod,
           sls_product_lookup pnumb,
           sls_order_method_dim meth
   where mkt.order_day_key=sales.order_day_key
     and sales.product_key=prod.product_key
     and prod.product_number=pnumb.product_number
     and pnumb.product_language='EN'
     and meth.order_method_key=sales.order_method_key;
```

```
1 create view myview as
2   select product_name, sales.product_key, mkt.quantity,
3         sales.order_day_key, sales.sales_order_key, order_method_en
4   from
5     mrk_promotion_fact mkt,
6       sls_sales_fact sales,
7       sls_product_dim prod,
8       sls_product_lookup pnumb,
9       sls_order_method_dim meth
10  where mkt.order_day_key=sales.order_day_key
11    and sales.product_key=prod.product_key
12    and prod.product_number=pnumb.product_number
13    and pnumb.product_language='EN'
14    and meth.order_method_key=sales.order_method_key;
15
```

- 2. Now, query the view.

```
select * from myview
  order by product_key asc, order_day_key asc
  fetch first 20 rows only;
```

- 3. Inspect the results:

The screenshot shows a database query results interface. At the top, there are navigation links: 'Discover' (with a magnifying glass icon), '?', and a user profile icon. Below that, tabs for 'Script Library' and 'Result History' are visible. The main area displays a query result set titled 'Result - 09/16/20 20:17:32'. The query is: 'select \* from myview order by product\_key asc, order\_day\_key asc fetch ...'. The run time is listed as '5.110s'. The results are presented in a table with the following columns: PRODUCT\_NAME, PRODUCT\_KEY, QUANTITY, ORDER\_DAY\_KEY, SALES\_ORDER\_KEY, and ORDER\_METHOD\_EN. The data shows multiple entries for 'TrailChef Water Bag' with various quantities (e.g., 1107, 714, 634, 891) and order day keys (e.g., 20040112). The 'ORDER\_METHOD\_EN' column indicates methods like 'Sales visit' and 'E-mail'. A 'Search' bar and navigation icons are at the top of the table. A 'Show Less' link is at the bottom right of the table.

PRODUCT_NAME	PRODUCT_KEY	QUANTITY	ORDER_DAY_KEY	SALES_ORDER_KEY	ORDER_METHOD_EN
TrailChef Water Bag	30001	1107	20040112	177811	Sales visit
TrailChef Water Bag	30001	1107	20040112	195305	Sales visit
TrailChef Water Bag	30001	1107	20040112	194990	E-mail
TrailChef Water Bag	30001	714	20040112	195305	Sales visit
TrailChef Water Bag	30001	714	20040112	177811	Sales visit
TrailChef Water Bag	30001	634	20040112	195305	Sales visit
TrailChef Water Bag	30001	714	20040112	194990	E-mail
TrailChef Water Bag	30001	891	20040112	195305	Sales visit
TrailChef Water Bag	30001	634	20040112	177811	Sales visit
TrailChef Water Bag	30001	1350	20040112	195305	Sales visit
TrailChef Water Bag	30001	634	20040112	194990	E-mail
TrailChef Water Bag	30001	766	20040112	195305	Sales visit
TrailChef Water Bag	30001	891	20040112	177811	Sales visit
TrailChef Water Bag	30001	663	20040112	195305	Sales visit
TrailChef Water Bag	30001	891	20040112	194990	E-mail
TrailChef Water Bag	30001	495	20040112	195305	Sales visit
TrailChef Water Bag	30001	1350	20040112	177811	Sales visit
TrailChef Water Bag	30001	1260	20040112	195305	Sales visit
TrailChef Water Bag	30001	1350	20040112	194990	E-mail
TrailChef Water Bag	30001	965	20040112	195305	Sales visit

## Part 6: Populating a table with 'INSERT INTO ... SELECT'

You can populate a table with data based on the results of a query by using Db2 Big SQL. In this part, you use an INSERT INTO...SELECT statement to retrieve data from multiple tables and insert that data into another table. The INSERT INTO...SELECT statement exploits the machine resources of your cluster because Db2 Big SQL can parallelize both read (SELECT) and write (INSERT) operations.

The file `6_INSERT INTO SELECT statements.sql` file is located in the folder `/home/bigsql/labfiles/bigsql/BIG_SQL_Data_Analysis`.

- 1. Run the following statement to create a sample table named `sales_report`:

- Create a sample `sales_report` table

```
CREATE HADOOP TABLE sales_report
(
product_key      INT NOT NULL,
product_name     VARCHAR(150),
quantity         INT,
order_method_en  VARCHAR(90)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

- 2. Now, populate the newly created table with results from a query that joins data from multiple tables.

- Populate the `sales_report` data with results from a query

```
INSERT INTO sales_report
SELECT sales.product_key, pnumb.product_name, sales.quantity,
meth.order_method_en
FROM
sls_sales_fact sales,
sls_product_dim prod,
sls_product_lookup pnumb,
sls_order_method_dim meth
WHERE
pnumb.product_language='EN'
AND sales.product_key=prod.product_key
AND prod.product_number=pnumb.product_number
AND meth.order_method_key=sales.order_method_key
and sales.quantity > 1000;
```

- 3. Verify that the previous query ran successfully by running the following query.

- Total number of rows should be 14441

```
select count(*) from sales_report;
```

## **Part 7: Working with external tables**

The previous tasks in this exercise caused Db2 Big SQL to store tables in a default location in the Hive warehouse. Db2 Big SQL also supports the concept of an externally managed table, for example, a table that is created over a user directory that resides outside the Hive warehouse. This user directory contains all the table's data in files.

In this part, you create a DFS directory, upload data into it, and then create a Db2 Big SQL table over this directory. To satisfy queries, Db2 Big SQL looks in the user directory that is specified when

you created the table and considers all files in that directory to be the table's contents. After the table is created, you query that table.

- 1. Create directories in your distributed file system for the source data files by using PuTTY. Ensure public read/write access to these directories.

```
hdfs dfs -mkdir /user/<username>/bigsq1_lab
hdfs dfs -mkdir /user/<username>/bigsq1_lab/sls_product_dim
hdfs dfs -chmod -R 777 /user/<username>/bigsq1_lab
```

- 2. Upload the source data files into their respective DFS directories. Change the local and DFS directories information to match your environment.

```
hdfs dfs -copyFromLocal
/usr/ibmpacks/bigsq1/6.0.0.0/bigsq1/samples/data/GOSALES DW.SLS_PRODUCT_DIM.t
xt /user/<username>/bigsq1_lab/sls_product_dim/SLS_PRODUCT_DIM.txt
```

- 3. List the contents of the DFS directories into which you copied the files to validate your work.

```
hdfs dfs -ls /user/<username>/bigsq1_lab/sls_product_dim
```

```
[student0003@dataengineer ~]$ hdfs dfs -ls /user/student0003/bigsq1_lab/sls_product_dim
Found 1 items
-rw-r--r-- 3 student0003 hdfs 24089 2020-09-16 19:35 /user/student0003/bigsq1_lab/sls_product_dim/SLS_PRODUCT_DIM
[student0003@dataengineer ~]$ █
```

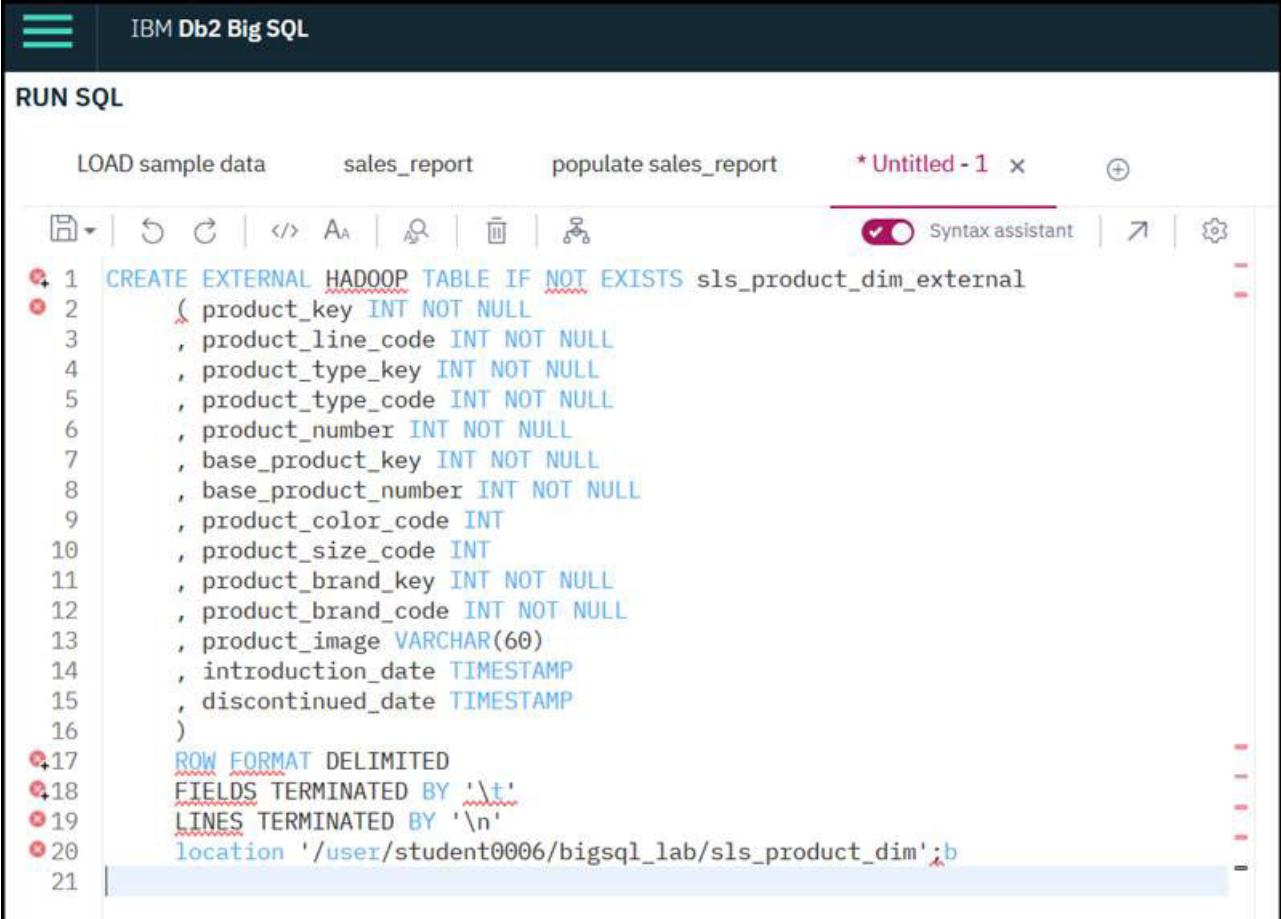
- 4. From your query execution environment, for example, JSqsh or the Db2 Big SQL Console, create an external Db2 Big SQL table for the sales product dimension, **sls\_product\_dim\_external**.

Note that the LOCATION clause in each statement references the DFS directory into which you copied the sample data.

The file **8\_External\_tables.sql** is located in the folder  
**/home/bigsq1/labfiles/bigsq1/BIG\_SQL\_Data\_Analysis**.

- Product dimension table that is stored in a DFS directory external to Hive

```
CREATE EXTERNAL HADOOP TABLE IF NOT EXISTS
  sls_product_dim_external
  (
    product_key INT NOT NULL
    , product_line_code INT NOT NULL
    , product_type_key INT NOT NULL
    , product_type_code INT NOT NULL
    , product_number INT NOT NULL
    , base_product_key INT NOT NULL
    , base_product_number INT NOT NULL
    , product_color_code INT
    , product_size_code INT
    , product_brand_key INT NOT NULL
    , product_brand_code INT NOT NULL
    , product_image VARCHAR(60)
    , introduction_date TIMESTAMP
    , discontinued_date TIMESTAMP
  )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
location '/user/<username>/bigsqllab/sls_product_dim';
```



The screenshot shows the IBM Db2 Big SQL interface. The title bar says "IBM Db2 Big SQL". Below it, a toolbar has buttons for "LOAD sample data", "sales\_report", "populate sales\_report", and a new tab titled "\* Untitled - 1". The main area is a code editor with the following SQL script:

```

1 CREATE EXTERNAL HADOOP TABLE IF NOT EXISTS sls_product_dim_external
2   ( product_key INT NOT NULL
3   , product_line_code INT NOT NULL
4   , product_type_key INT NOT NULL
5   , product_type_code INT NOT NULL
6   , product_number INT NOT NULL
7   , base_product_key INT NOT NULL
8   , base_product_number INT NOT NULL
9   , product_color_code INT
10  , product_size_code INT
11  , product_brand_key INT NOT NULL
12  , product_brand_code INT NOT NULL
13  , product_image VARCHAR(60)
14  , introduction_date TIMESTAMP
15  , discontinued_date TIMESTAMP
16  )
17 ROW FORMAT DELIMITED
18 FIELDS TERMINATED BY '\t'
19 LINES TERMINATED BY '\n'
20 location '/user/student0006/bigsql_lab/sls_product_dim';\b
21

```



### Note

You can ignore the syntax errors that are shown in the figure and continue to the next step.

---

5. Query the table.

```

select product_key, introduction_date from sls_product_dim_external
where discontinued_date is not null
fetch first 20 rows only;

```

6. Inspect the results.

The screenshot shows a database query results interface. At the top, there are navigation links: 'Discover' (with a magnifying glass icon), '?', and a user profile icon. Below the header, there are tabs for 'Script Library' and 'Result History'. The main area displays a query result set titled 'Result - 09/16/2020 21:18:01'. The query is: 'select product\_key, introduction\_date from sls\_product\_dim\_external whe...'. The run time is listed as '3.663s'. The result set is labeled 'Result set 1' and includes a 'Search' bar with a magnifying glass icon and sorting icons for up and down. The table has two columns: 'PRODUCT\_KEY' and 'INTRODUCTION\_DATE'. The data shows multiple rows of product keys, all corresponding to the same introduction date of '2003-01-01 00:00:00.0'. The last row, '30217', is highlighted with a light gray background.

PRODUCT_KEY	INTRODUCTION_DATE
30134	2003-01-01 00:00:00.0
30139	2003-01-01 00:00:00.0
30143	2003-01-01 00:00:00.0
30146	2003-01-01 00:00:00.0
30147	2003-01-01 00:00:00.0
30154	2003-01-01 00:00:00.0
30156	2003-01-01 00:00:00.0
30159	2003-01-01 00:00:00.0
30160	2003-01-01 00:00:00.0
30162	2003-01-01 00:00:00.0
30169	2003-01-01 00:00:00.0
30174	2003-01-01 00:00:00.0
30178	2003-01-01 00:00:00.0
30186	2003-01-01 00:00:00.0
30199	2003-01-01 00:00:00.0
30202	2003-01-01 00:00:00.0
30204	2003-01-01 00:00:00.0
30205	2003-01-01 00:00:00.0
30213	2003-01-01 00:00:00.0
30217	2003-01-01 00:00:00.0

[Show Less](#)

## End of exercise

## Exercise review and wrap-up

In this exercise, you started by creating and dropping a simple Db2 Big SQL table. Then, you created multiple Db2 Big SQL tables with a variety of data types and loaded the tables with data. You also worked with views, external tables, and other methods of creating Db2 Big SQL tables.

# Exercise 3. Querying Db2 Big SQL tables

## Estimated time

01:00

## Overview

In this exercise, you experiment with advanced SQL queries. Then, you explore the Db2 Big SQL ARRAY type. You also create a user-defined function (UDF) and write queries that call the UDF. Finally, you store data in an alternative file format (Parquet).

## Objectives

After completing this exercise, you will be able to:

- Run more advanced Db2 Big SQL queries.
- Work with ARRAY data type.
- Create user-defined functions (UDFs).
- Store data in popular formats for Hadoop environments such as Parquet.

## Introduction

With Db2 Big SQL you can create common table expressions, aggregate functions, and ranking, which allows you to better analyze your data and make decisions.

Db2 Big SQL supports the ARRAY data type, which is used to define a data structure for homogeneous objects. Db2 Big SQL ARRAY data type can be used in simple and nested forms.

User-defined functions (UDFs) are extensions or additions to the existing built-in functions of the Db2 Big SQL language. The Db2 Big SQL scalar functions are implemented as static methods on a class. UDFs can be invoked in queries.

File formats play a significant role in Hadoop environments since they help improve the performance of storing and processing huge amounts of data. Db2 Big SQL provides built-in support for numerous file formats, including Parquet, ORC, Avro, text, and more. Additionally, Db2 Big SQL offers various compression mechanisms for certain file formats, which provide added performance benefits.

In this exercise, you experiment with the capabilities listed in this section.

## Requirements

- Access to the values provided by your instructor and that you recorded in the table in “Exercise 1. Connecting to the IBM Db2 Big SQL server”.
  - Completed Exercise 1, “Part 1. Adding the Hadoop cluster hostname to the hosts file”.
- 



### Note

The hostnames that are shown in the screen captures of this exercise might be different from the one in your hands-on environment.

---

- PuTTY SSH client installed in your workstation.

## Exercise instructions

In this exercise, you complete the following tasks:

- \_\_ 1. Connect to Db2 Big SQL.
- \_\_ 2. Query data with Db2 Big SQL.
- \_\_ 3. Work with the ARRAY type.
- \_\_ 4. Work with Db2 Big SQL functions.
- \_\_ 5. Store data in an alternative file format (Parquet).

### **Part 1: Connecting to Db2 Big SQL**

Db2 Big SQL queries are run by the Db2 Big SQL server on your cluster against data in your cluster. To run Db2 Big SQL queries, your environment must be associated with a Db2 Big SQL server that is part of the Db2 Big SQL service. To run Db2 Big SQL queries, you must first connect to a Db2 Big SQL server. You can run Db2 Big SQL queries from JSqsh.

In this part, you start the JSqsh shell and connect to the Db2 Big SQL server by using the connection that you set up in “Exercise 1. Connecting to the IBM Db2 Big SQL server”.

- \_\_ 1. Open **PuTTY** to connect to the VM. Enter the <hostname>, <username>, and <password>.
- \_\_ 2. Start the **JSqsh** shell:  

```
/usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh
```
- \_\_ 3. From the **JSqsh** shell, connect to your Db2 Big SQL server by using the connection **bysql** that you created in “Exercise 1. Connecting to the IBM Db2 Big SQL server”.  

```
\connect bysql
```

If prompted for a password, enter your student’s password <password>.
- \_\_ 4. Run **\tables** to display the list of all the tables that you created in prior tasks. The results are similar to the following output.

```
[dataengineer.ibm.com] [student0003] 1> \tables
+-----+-----+
| TABLE_SCHEMA | TABLE_NAME           | TABLE_TYPE |
+-----+-----+
| STUDENT0003  | GO_REGION_DIM          | TABLE      |
| STUDENT0003  | MRK_PROMOTION_FACT     | TABLE      |
| STUDENT0003  | SALES_REPORT            | TABLE      |
| STUDENT0003  | SLS_ORDER_METHOD_DIM    | TABLE      |
| STUDENT0003  | SLS_PRODUCT_BRAND_LOOKUP| TABLE      |
| STUDENT0003  | SLS_PRODUCT_DIM          | TABLE      |
| STUDENT0003  | SLS_PRODUCT_DIM_EXTERNAL| TABLE      |
| STUDENT0003  | SLS_PRODUCT_LINE_LOOKUP | TABLE      |
| STUDENT0003  | SLS_PRODUCT_LOOKUP       | TABLE      |
| STUDENT0003  | SLS_SALES_FACT           | TABLE      |
| STUDENT0003  | MYVIEW                   | VIEW       |
+-----+-----+
```

## Part 2: Querying data with Db2 Big SQL

In this part, you experiment with some more advanced SQL queries. You query the data that you loaded previously into the Db2 Big SQL tables.

- 1. Join data from multiple tables to return the product name, quantity, and order method of goods that were sold. For simplicity, limit the number of returned rows to 20. To achieve this goal, run the following query.



### Note

This SQL statement can also be found in the file `4_SELECT statements - querying data.sql` that is located in the folder `/home/bysql/labfiles/bysql/BIG_SQL_Data_Analysis`.

```
--Fetch the product name, quantity, and order method of products sold.
--
--Query 1
SELECT pnumb.product_name, sales.quantity,
meth.order_method_en
FROM
sls_sales_fact sales,
sls_product_dim prod,
sls_product_lookup pnumb,
sls_order_method_dim meth
WHERE
pnumb.product_language='EN'
AND sales.product_key=prod.product_key
AND prod.product_number=pnumb.product_number
AND meth.order_method_key=sales.order_method_key
fetch first 20 rows only;
```

Data from four tables is used to drive the results of this query (see the tables referenced in the FROM clause). Relationships between these tables are resolved through three `join` predicates that are specified as part of the WHERE clause. The query relies on three `equi-joins` to filter data from the referenced tables.

Predicates such as `prod.product_number=pnumb.product_number` help to narrow the results to product numbers that match in two tables.

For improved readability, this query uses aliases in the SELECT and FROM clauses to refer to tables. For example, `pnumb.product_name` refers to "pnumb," which is the alias for the table `gosalesdw.sls_product_lookup`.

After it is defined in the FROM clause, an alias can be used in the WHERE clause so that you do not need to repeat the complete table name.

The use of the predicate and `pnumb.product_language='EN'` helps to further narrow the result to only English output. This database contains thousands of rows of data in various languages, so restricting the language provides some optimization.

The following figure shows the results from JSqsh, but you can use the Db2 Big SQL Console to run this query.



### Note

The results of the queries might be different for you.

PRODUCT_NAME	QUANTITY	ORDER_METHOD_EN
EverGlow Lamp	484	Sales visit
Course Pro Putter	587	Telephone
Blue Steel Max Putter	214	Telephone
Course Pro Gloves	576	Telephone
Glacier Deluxe	129	Sales visit
BugShield Natural	1776	Sales visit
Sun Shelter 15	1822	Sales visit
Compact Relief Kit	412	Sales visit
Hailstorm Titanium Woods Set	67	Sales visit
Canyon Mule Extreme Backpack	97	E-mail
TrailChef Canteen	1172	Telephone
TrailChef Cook Set	591	Telephone
TrailChef Deluxe Cook Set	338	Telephone
Star Gazer 3	97	Telephone
Hibernator	364	Telephone
Hibernator Camp Cot	234	Telephone
Canyon Mule Cooler	603	Telephone
Firefly 4	232	Telephone
EverGlow Single	450	Telephone
EverGlow Kerosene	257	Telephone

20 rows in results(first row: 10.629s; total: 10.647s)

Modify the query to restrict the order method to one type: those that involve a Sales visit. To do so, add the following query predicate just before the **FETCH**.

```
--Fetch the product name, quantity, and order method
--of products sold through sales visits.
--Query 2
SELECT pnumb.product_name, sales.quantity,
meth.order_method_en
FROM
  sls_sales_fact sales,
  sls_product_dim prod,
  sls_product_lookup pnumb,
  sls_order_method_dim meth
WHERE
  pnumb.product_language='EN'
  AND sales.product_key=prod.product_key
  AND prod.product_number=pnumb.product_number
  AND meth.order_method_key=sales.order_method_key
  AND order_method_en='Sales visit'
  FETCH FIRST 20 ROWS ONLY;
```

- \_\_\_ 2. Inspect the results.

PRODUCT_NAME	QUANTITY	ORDER_METHOD_EN
Hailstorm Titanium Woods Set	50	Sales visit
Glacier Deluxe	129	Sales visit
BugShield Natural	1776	Sales visit
Sun Shelter 15	1822	Sales visit
Compact Relief Kit	412	Sales visit
Hailstorm Titanium Woods Set	67	Sales visit
TrailChef Double Flame	205	Sales visit
TrailChef Utensils	950	Sales visit
Star Lite	334	Sales visit
Star Gazer 2	205	Sales visit
Hibernator Lite	459	Sales visit
Firefly Extreme	128	Sales visit
EverGlow Double	36	Sales visit
Mountain Man Deluxe	129	Sales visit
Polar Extreme	23	Sales visit
Edge Extreme	286	Sales visit
Bear Edge	246	Sales visit
Seeker 50	154	Sales visit
Glacier GPS Extreme	123	Sales visit
BugShield Spray	1266	Sales visit

20 rows in results(first row: 0.899s; total: 0.905s)

- \_\_\_ 3. To find out which sales method of all the methods has the greatest quantity of orders, include a GROUP BY clause (group by `p11.product_line_en, md.order_method_en`). In addition, invoke the SUM aggregate function (`sum(sf.quantity)`) to total the orders by product and method. Finally, this query cleans up the output by using aliases (for example, AS Product) to provide a more readable column header.

## --Query 3

```

SELECT pll.product_line_en AS Product,
       md.order_method_en AS Order_method,
       sum(sf.QUANTITY) AS total
  FROM
    sls_order_method_dim AS md,
    sls_product_dim AS pd,
    sls_product_line_lookup AS pll,
    sls_product_brand_lookup AS pbl,
    sls_sales_fact AS sf
 WHERE
    pd.product_key = sf.product_key
  AND md.order_method_key = sf.order_method_key
  AND pll.product_line_code = pd.product_line_code
  AND pbl.product_brand_code = pd.product_brand_code
 GROUP BY pll.product_line_en, md.order_method_en;

```

- 4. Inspect the results, which should contain 35 rows. Partial results are shown in the following figure.

PRODUCT	ORDER_METHOD	TOTAL
Camping Equipment	E-mail	1413084
Camping Equipment	Fax	413958
Camping Equipment	Mail	348058
Camping Equipment	Sales visit	2899754
Camping Equipment	Special	203528
Camping Equipment	Telephone	2792588
Camping Equipment	Web	19230179
Golf Equipment	E-mail	333300
Golf Equipment	Fax	102651
Golf Equipment	Mail	80432
Golf Equipment	Sales visit	263788
Golf Equipment	Special	38585
Golf Equipment	Telephone	601506
Golf Equipment	Web	3693439
Mountaineering Equipment	E-mail	199214
Mountaineering Equipment	Fax	292408
Mountaineering Equipment	Mail	81259
Mountaineering Equipment	Sales visit	1041237
Mountaineering Equipment	Special	93856
Mountaineering Equipment	Telephone	549811

- 5. Type **quit** to exit the JSqsh shell.

**Part 3: Working with the ARRAY type**

In this task, you explore the Db2 Big SQL ARRAY type, which stores values of the same data type as an ordered collection in a single column of a table. First, you create a variation of the `sls_product_line_lookup` table that you created earlier.

Your new table includes two columns: an integer column for the product line code and an array column of varying-length character strings for product line descriptions in three different languages. Then, you load data into this table from a file and query the table.

The file **Array\_Data.txt** is located in the folder  
**/home/bigsq1/labfiles/bigsq1/BIG\_SQL\_Data\_Analysis**.

- \_\_\_ 1. Copy the file **Array\_Data.txt** into a new file **product\_line\_array.csv** in your home directory.

```
cp /home/bigsq1/labfiles/bigsq1/BIG_SQL_Data_Analysis/Array_Data.txt
/home/<username>/product_line_array.csv
```

For simplicity, this sample data represents a subset of the data that is contained in the table **sls\_product\_line\_lookup**.

- \_\_\_ 2. Inspect the contents of your file by using the command line. Ignore any special language-specific characters that do not display properly from the shell.

```
cat /home/<username>/product_line_array.csv
```

The following figure shows the results.

```
[student0006@dataengineer ~]$ cp /home/bigsq1/labfiles/bigsq1/BIG_SQL_Data_Analysis/Array_Data.txt /home/student0006/product_line_array.csv
[student0006@dataengineer ~]$ cat /home/student0006/product_line_array.csv
991|Camping Equipment,Campingausstattung,Matériel de camping
992|Mountaineering Equipment,Bergsteigerausstattung,Matériel de montagne
993|Personal Accessories,Accessoires,Accessoires personnels
```

- \_\_\_ 3. Place a copy of this new csv file on HDFS in the directory **/user/<username>/sampledata/data**, by running the following command:

```
hadoop fs -copyFromLocal /home/<username>/product_line_array.csv
/user/<username>/sampledata/data
```

- \_\_\_ 4. List the contents of the HDFS directory:

```
hadoop fs -ls /user/<username>/sampledata/data
```

Your new file **product\_line\_array.csv** should be in the list. The following output shows partial results, which include the file **product\_line\_array.csv**.

```
[student0001@dataengineer ~]$ hadoop fs -ls /user/student0001/sampledata/data
Found 70 items
[partial results]

-rwxr--r-x  3 student0001 hdfs      11953 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_PRODUCT_SIZE_LOOKUP.txt
-rwxr--r-x  3 student0001 hdfs      6617 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_PRODUCT_TYPE_LOOKUP.txt
-rwxr--r-x  3 student0001 hdfs     1037029 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_RTL_DIM.txt
-rwxr--r-x  3 student0001 hdfs    51258157 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_SALES_FACT.txt
-rwxr--r-x  3 student0001 hdfs   11133505 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_SALES_ORDER_DIM.txt
-rwxr--r-x  3 student0001 hdfs  10614522 2020-10-05 23:18 /user/student0001/sampledata/data/GOSALES DW.SLS_SALES_TARG_FACT.txt
-rw-r--r--  3 student0001 hdfs      317 2020-10-10 19:07 /user/student0001/sampledata/data/product_line_array.csv
[student0001@dataengineer ~]$
```

- \_\_\_ 5. Launch your query execution tool, for example JSqsh, and establish a connection to your Db2 Big SQL server.

```
/usr/ibmpacks/common-utils/current/jsqsh/bin/jsqsh
\connect bigsql
```

- \_\_\_ 6. Create the table **sls\_product\_line\_lookup\_array**. Specify a single ARRAY column for all of the product descriptions:

```
create.hadoop table if not exists sls_product_line_lookup_array
(product_line_code      INT NOT NULL,
description            ARRAY<VARCHAR(50)>);
```

The expected result is similar to the following output.

```
0 rows affected (total: 0.930s)
```

- 7. Load the sample data from your file `product_line_array.csv` into this table.

```
LOAD HADOOP USING FILE URL
'/user/<username>/sampledata/data/product_line_array.csv' WITH SOURCE
PROPERTIES ('field.delimiter'='|', 'collection.items.delimiter'=',') INTO
TABLE sls_product_line_lookup_array OVERWRITE;
```



### Important

**LONG RUNNING commands.** When a single user submits a LOAD command, it usually completes in about 1 minute. In the current class environment, when tens of students run commands at the same time, more wait time is to be expected. In our tests, the maximum wait time experienced was approximate 5 minutes. Be patient and do not close or break your session.

- 8. Inspect the informational message that is displayed after the LOAD operation completes. Verify that the number of rows loaded (processed) into this table is 5 and that 0 rows were skipped.

```
[dataengineer.ibm.com] [student0001] 1> LOAD HADOOP USING FILE URL '/user/student0001/sampledata/data/product_line_array.csv' WITH SOURCE PROPERTIES ('field.delimiter'='|',
'collection.items.delimiter'=',') INTO TABLE sls_product_line_lookup_array OVERWRITE;
WARN [State: 0    ][Code: 5108]: Loading of data to a Hadoop table or processing of data in an external table completed. Number of rows processed: "5". Number of source records: "5". If the
source was a file, number of skipped lines: "0". Number of rejected source records: "0". Job or file identifier: "job_1601999620872_0013"., SQLCODE=5108, SQLSTATE=0    , DRIVER=3.72.46
0 rows affected (total: 1m4.940s)
```

- 9. Query the table. Retrieve only the product line code, English description, and German description for product lines with English descriptions that begin with the word "Personal":

```
select product_line_code, description[1] as English, description[2] as
German from sls_product_line_lookup_array where description[1] like
'Personal%';
```

Note that this query effectively "flattens" the array's content by identifying specific elements of interest and including each in a separate column of the result set. In this case, the query retrieves the first element of the array (which contains the product line's English description) and the second element of the array (which contains the product line's German description).

Inspect the results.

PRODUCT_LINE_CODE	ENGLISH	GERMAN
993	Personal Accessories	Accessoires

1 row in results(first row: 0.237s; total: 0.238s)

- 10. Attempt to query the table by using improper syntax for the array column:

```
select * from sls_product_line_lookup_array;
```

Inspect the error message that is returned. When you include a complex column in the SELECT list or WHERE clause of a query, you must specify the individual element or field of interest.

```
[dataengineer.ibm.com] [student0003] 1> select * from sls_product_line_lookup_array;
SQL Exception(s) Encountered:
[State: 428H2][Code: -20441]: A "ARRAY" data type is not supported in the context where it is being used.. SQLCODE=-20441, SQLSTAT
E=428H2, DRIVER=3.72.46
[State: 56098][Code: -727]: An error occurred during implicit system action type "2". Information returned for the error includes
SQLCODE "-20441", SQLSTATE "428H2" and message tokens "ARRAY".. SQLCODE=-727, SQLSTATE=56098, DRIVER=3.72.46
```

## **Part 4: Working with Db2 Big SQL functions**

Db2 Big SQL enables users to create their own SQL functions that can be invoked in queries. User-defined functions (UDFs) promote code re-use and reduce query complexity. They can be written to return a single (scalar) value or a result set (table). Programmers can write UDFs in SQL or any supported programming languages (such as Java and C). For simplicity, this task focuses on SQL UDFs.

Db2 Big SQL provides many built-in functions to perform common computations. Often, organizations need to perform some customized or complex operation on their data that is beyond the scope of any built-in-function. Db2 Big SQL allows users to embed their customized business logic inside a user-defined function (UDF) and write queries that call these UDFs.

- \_\_\_ 1. Within JSqsh reset the default SQL terminator character to “@”:

```
\set terminator = @;
```

Because some of the UDFs that you will develop involve multiple SQL statements, you must reset the JSqsh default termination character so that the semicolon that follows each SQL statement in your UDF is not interpreted as the end of the CREATE FUNCTION statement for your UDF.

- \_\_\_ 2. Confirm that the terminator is reset.

```
\set
```

readline	JLine
scale	5
server	dataengineer.ibm.com
shell	/bin/sh,-c,?
show_exclass	false
show_meta	false
show_stack	false
style	perfect
<b>terminator</b>	<b>@</b>
timeout	0
timer	true
user	student0003
version	6.0.0.3
width	130
window_size	600x400

Inspect the output from the command as shown in the figure and verify that the terminator property is set to @.

Next, you create a scalar SQL UDF to compute the final price of a particular item that was sold. Your UDF requires several input parameters:

unit sale price: Price of one item  
 quantity: Number of units of this item being sold in this transaction  
 % discount: Discount on the item (computed before tax)  
 % sales-tax: Sales tax (computed after discount)

Your UDF returns a single value: The final price of the item.

After you create and register the UDF, you invoke it by using some test values to ensure that it behaves correctly. Afterward, you invoke the UDF in a query, passing the values from columns in a table as input to your function.

- 3. Create a UDF named `new_final_price` in the default schema.

```
CREATE OR REPLACE FUNCTION new_final_price
(
  quantity INTEGER,
  unit_sale_price DOUBLE,
  discount_in_percent DOUBLE,
  sales_tax_in_percent DOUBLE
)
RETURNS DOUBLE
LANGUAGE SQL
RETURN (quantity * unit_sale_price) * DOUBLE(1 - discount_in_percent / 100.0)
* DOUBLE(1 + sales_tax_in_percent / 100.0) @
```

Review the logic of this function. The first line creates the function, which is defined to take four input parameters: `quantity`, `unit_sale_price`, `discount_in_percent`, and `sales_tax_in_percent`.

- The RETURNS clause indicates that a single (scalar) value of type DOUBLE is returned.
- The function's language is SQL.
- Finally, the last two lines include the function's logic, which performs the necessary arithmetic operations to calculate the final sales price of an item.

The expected result is as follows.

```
0 rows affected (total: 0.046s)
```

- 4. After you create the function, test it by using some sample values. A simple way to test the function is with the VALUES clause shown here:

```
VALUES new_final_price (1, 10, 20, 8.75)@
```

-----+
1
-----+
8.70000
-----+
1 row in results(first row: 0.011s; total: 0.015s)

- \_\_\_ 5. Use the UDF in a query to compute the final price for items listed in sales transactions in the SLS\_SALES\_FACT table.

```
SELECT sales_order_key, quantity, unit_sale_price, new_final_price(quantity,
unit_sale_price, 20, 8.75) as final_price
FROM sls_sales_fact
ORDER BY sales_order_key
FETCH FIRST 10 ROWS ONLY@
```

Note that this query uses values from two columns in the table as input for the quantity and unit price and two user-supplied values as input for the discount rate and sales tax rate.

- \_\_\_ 6. Inspect the results of the query.

SALES_ORDER_KEY	QUANTITY	UNIT_SALE_PRICE	FINAL_PRICE
100001	256	33.69000	7503.43680
100002	92	102.30000	8188.09200
100003	162	111.31000	15688.03140
100004	172	38.90000	5820.99600
100005	74	334.43000	21530.60340
100006	90	75.84000	5938.27200
100007	422	6.00000	2202.84000
100008	3252	6.51000	18418.35240
100009	1107	5.76000	5547.39840
100010	88	124.72000	9548.56320

10 rows in results(first row: 1.191s; total: 1.197s)

- \_\_\_ 7. Now invoke your UDF in the WHERE clause of a query. Note that scalar UDFs can be included anywhere in an SQL statement that a scalar value is expected. This query is similar to your previous query except that it includes a WHERE clause to restrict the result set to items with a file price greater than 7000.

```
SELECT sales_order_key, quantity, unit_sale_price,
new_final_price(quantity, unit_sale_price, 20, 8.75) as final_price
FROM sls_sales_fact
WHERE new_final_price(quantity, unit_sale_price, 20, 8.75) > 7000
ORDER BY sales_order_key
FETCH FIRST 10 ROWS ONLY@
```

- \_\_\_ 8. Note that your results no longer include rows with items that are priced 7000 or lower.

```
+-----+-----+-----+-----+
| SALES_ORDER_KEY | QUANTITY | UNIT_SALE_PRICE | FINAL_PRICE |
+-----+-----+-----+-----+
| 100001 | 256 | 33.69000 | 7503.43680 |
| 100002 | 92 | 102.30000 | 8188.09200 |
| 100003 | 162 | 111.31000 | 15688.03140 |
| 100005 | 74 | 334.43000 | 21530.60340 |
| 100008 | 3252 | 6.51000 | 18418.35240 |
| 100010 | 88 | 124.72000 | 9548.56320 |
| 100012 | 354 | 83.78000 | 25802.56440 |
| 100013 | 261 | 344.22000 | 78162.03540 |
| 100014 | 139 | 541.65000 | 65501.73450 |
| 100015 | 279 | 120.64000 | 29282.94720 |
+-----+-----+-----+-----+
10 rows in results(first row: 0.885s; total: 0.889s)
```

- \_\_\_ 9. Set terminator back to semi-colon ":"

```
\set terminator = ;;
```

## **Part 5: Storing data in an alternative file format (Parquet)**

Until now, you instructed Db2 Big SQL to use the TEXTFILE format for storing data in the tables that you created. This format is easy to read (both by people and most applications), as data is stored in a delimited form with one record per line and new lines separate individual records. It is also the default format for Db2 Big SQL tables.

However, if you prefer to use a different file format for data in your tables, Db2 Big SQL supports several formats that are popular in the Hadoop environment, including Avro, sequence files, RC (record columnar) and Parquet.

While it is beyond the scope of this exercise to explore these file formats, in this part, you learn how you can override the default Db2 Big SQL file format to use another format, in this case, Parquet. Parquet is a columnar storage format for Hadoop. It is popular because of its support for efficient compression and encoding schemes. For more information about Parquet, see <http://parquet.io/>

The file `7_Parquet statements.sql` is located in the folder  
`/home/bigsq1/labfiles/bigsq1/BIG_SQL_Data_Analysis`.

- \_\_\_ 1. Create a table named `big_sales_parquet`.

```
CREATE HADOOP TABLE IF NOT EXISTS big_sales_parquet
(
product_key      INT NOT NULL,
product_name     VARCHAR(150),
quantity         INT,
order_method_en  VARCHAR(90)
)
STORED AS parquetfile;
```

With the exception of the final line (which specifies the PARQUETFILE format), all aspects of this statement should be familiar to you by now.

The expected result is as follows.

```
0 rows affected (total: 0.481s)
```

- \_\_\_ 2. Populate this table with data based on the results of a query.

Note that this query joins data from four tables that you previously defined in Db2 Big SQL by using a TEXTFILE format. Db2 Big SQL automatically reformats the result set of this query into a Parquet format for storage.

```
INSERT into big_sales_parquet
SELECT sales.product_key, pnumb.product_name, sales.quantity,
meth.order_method_en
FROM
sls_sales_fact sales,
sls_product_dim prod,
sls_product_lookup pnumb,
sls_order_method_dim meth
WHERE
pnumb.product_language='EN'
AND sales.product_key=prod.product_key
AND prod.product_number=pnumb.product_number
AND meth.order_method_key=sales.order_method_key
and sales.quantity > 5500;
```

The expected result is as follows.

```
471 rows affected (total: 3.002s)
```

### 3. Query the table.

Note that your SELECT statement does not need to be modified in any way because of the underlying file format.

```
select * from big_sales_parquet;
```

### 4. Inspect the results.

The following figure shows a partial view of the results.

	30050   Granite Carabiner	7156   Web	
	30090   Single Edge	6216   Web	
	30050   Granite Carabiner	7655   Web	
	30090   Single Edge	6111   Web	
	30050   Granite Carabiner	5685   Web	
	30050   Granite Carabiner	5553   E-mail	
	30050   Granite Carabiner	7399   Web	
	30090   Single Edge	6271   Web	
	30090   Single Edge	6010   Web	
	30090   Single Edge	5887   Web	
	30050   Granite Carabiner	7164   Web	
	30050   Granite Carabiner	6843   Web	
	30090   Single Edge	6683   Web	
	30090   Single Edge	6047   Web	
	30050   Granite Carabiner	7843   Web	
	30090   Single Edge	6275   Web	
	30050   Granite Carabiner	6610   Web	
	30090   Single Edge	8241   Web	
<hr/>			

```
471 rows in results(first row: 0.423s; total: 0.490s)
```

Remember that parquet files are not human-readable.

### 5. Type **quit** to exit JSqsh

— 6. Navigate to the directory

/warehouse/tablespace/external/hive/<username>.db/big\_sales\_parquet in the HDFS if you want to see the actual content of the table.

```
hadoop fs -ls
```

```
/warehouse/tablespace/external/hive/<username>.db/big_sales_parquet
```

The expected result is as follows.

- -rw-rw-rw-+ 3 bigsql hadoop 3466 2020-10-04 03:51  
/warehouse/tablespace/external/hive/student0004.db/big\_sales\_parquet/i\_1601646788472\_-3  
05786563\_20201004035137244\_1.0

## End of exercise

## Exercise review and wrap-up

In this exercise, you experimented with advanced SQL queries. Then, you explored Db2 Big SQL ARRAY type. You also created a user-defined function (UDF) and wrote queries that call the UDF. Finally, you stored data in an alternative file format (Parquet).

# Exercise 4. Configuring authorizations of Db2 Big SQL objects



## Important

This exercise requires administration authority that is not granted to the students in this course. This exercise can be performed only by the instructor. A recorded demo of this exercise is available.

## Estimated time

00:45

## Overview

Authorization of the Db2 Big SQL objects is controlled at several levels:

Level 1: Controlling access with authorization in the distributed file system.

Level 2: Authorization with the GRANT command.

Level 3: Authorization at the row and column level.

Level 4: Controlling access by using VIEWS or STORED PROCEDURES.

In this exercise, you focus on authorization at the row and column level.

For more information about authorization of the Db2 Big SQL objects, see [Authorization of Db2 Big SQL objects](#).

## Objectives

After completing this exercise, you will be able to:

- Create roles, assign privileges to roles, and grant roles to users.
- Use column masking and row-based access control to restrict access to your data.

## Introduction

Row and column access (RCAC) provides an extra layer of security to the privileges. It controls access to a table at the row level, column level, or both. The row permission rule is in the form of an SQL search condition that describes the set of rows to which a user has access. The column mask rule is an SQL CASE expression that describes the column values that a user is permitted to see and under what conditions.

You do not need to change your application to take advantage of RCAC. This is an important security advantage in using Db2 Big SQL because the security administrator controls the access,

not the SQL application. But, you should be aware that queries do not see the whole picture in terms of the data in the table unless granted specific permission to do so.

## Requirements

- For information about the variables used in this exercise, refer to the table in Exercise 1. “Connecting to the IBM Db2 Big SQL server”.
- PuTTY SSH client installed in your workstation.

## Exercise instructions

In this exercise, you complete the following tasks:

- \_\_ 1. Set up the scenario.
- \_\_ 2. Create roles and assign privileges to the roles.
- \_\_ 3. Create and enforce the permissions.
- \_\_ 4. Grant roles to users and activate the row access control.
- \_\_ 5. Switch between users in Db2 Big SQL console.
- \_\_ 6. Use row access control.
- \_\_ 7. Shield data with column masking.

### ***Part 1: Setting up the scenario***

Create the OS users for the scenario. Db2 Big SQL relies on OS authentication. If you have an LDAP environment, define the users in LDAP.

- \_\_ 1. As the `root` user, run these commands in all nodes to add three users to the OS. In this course, only one node is available.

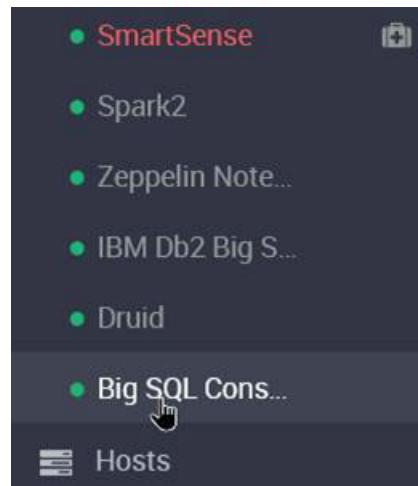
```
useradd branch_a
useradd branch_b
useradd cfo
```
- \_\_ 2. Set the passwords for the three users. If you get a message indicating that the password is weak, ignore it. This is just an example. Run the following commands and enter the user's password when prompted. In this example, the password is `password`.

```
passwd branch_a
passwd branch_b
passwd cfo
```

The results are similar to the following figure.

```
[root@dataengineer ~]# useradd branch_a
[root@dataengineer ~]# useradd branch_b
[root@dataengineer ~]# useradd cfo
[root@dataengineer ~]# passwd branch_a
Changing password for user branch_a.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@dataengineer ~]# passwd branch_b
Changing password for user branch_b.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@dataengineer ~]# passwd cfo
Changing password for user cfo.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
```

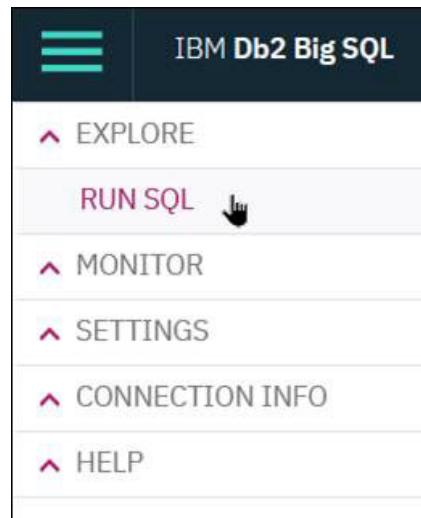
- \_\_\_ 3. Start the Ambari Web UI in a browser by entering the Ambari URL <hostname:8080>. Log in as <Ambari admin> with password <Ambari admin pwd>.
- \_\_\_ 4. Click the **Big SQL Console** service on the left navigation pane.



- \_\_\_ 5. If the Big SQL Console service is not already started, click **Actions > Start** to start it.
- \_\_\_ 6. Under **Quick Links**, click **Console UI**.

The screenshot shows the Ambari interface for a cluster named 'BDE\_Cluster'. On the left, a sidebar lists various services: Ranger, SmartSense, Spark2, Zeppelin Note, IBM Db2 Big S., Druid, and Big SQL Cons. The 'Big SQL Cons.' service is currently selected. The main panel displays the 'SUMMARY' tab for the 'Big SQL Console / Summary' service. It shows that the 'BIG SQL CONSOLE' component is 'Started'. A 'Quick Links' section on the right contains a link to 'Console UI'.

- \_\_\_ 7. Log in as **bigsq1** and <**bigsq1\_password**>
- \_\_\_ 8. Click the hamburger menu, then click **RUN SQL**.



- \_\_\_ 9. Click **Blank**.

The screenshot shows the 'RUN SQL' interface. At the top, it says 'Choose the way to create' and 'Open a script to edit'. Below this, there are two options: a red circle with a plus sign containing the text 'Blank' (which is highlighted with a cursor icon) and a downward arrow pointing to the text 'From file'.

- \_\_\_ 10. On the canvas, type or copy/paste this DDL. Overwrite the existing information is necessary.

```
create hadoop table hr.staff (
    empno int not null primary key,
    first_name varchar(20),
    salary decimal (9,2),
    branch_name varchar(10)
);
```

- \_\_\_ 11. Click **Run all** to run the statement.

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for Discover, Help, and User. Below the bar, it says "RUN SQL". The main area has a code editor with the following SQL statement:

```
* Untitled - 1
1 create hadoop table hr.staff (
2     empno int not null primary key,
3     first_name varchar(20),
4     salary decimal (9,2),
5     branch_name varchar(10)
6 );
7
```

Below the code editor is a button labeled "Run all" with a play icon. To its right is a checkbox labeled "Remember my last behavior". To the right of the code editor is a results panel titled "Result - 01/13/21 13:11:08". It shows the executed SQL statement and its result:

create hadoop table hr.staff ( ... )  
Run time: 0.447s  
Status: Success | Affected Rows: 0

- \_\_\_ 12. On the canvas, type or copy/paste this **insert** statement to populate the table.

```
insert into hr.staff values
(1,'Steve',rand()*30000,'Branch_B'),
(2,'Chris',rand()*30000,'Branch_A'),
(3,'Paula',rand()*30000,'Branch_A'),
(4,'Craig',rand()*30000,'Branch_B'),
(5,'Pete',rand()*30000,'Branch_A'),
(6,'Stephanie',rand()*30000,'Branch_B'),
(7,'Julie',rand()*30000,'Branch_B'),
(8,'Chrissie',rand()*30000,'Branch_A');
```

- \_\_\_ 13. Click **Run all** to run the statement.

The screenshot shows the IBM Db2 Big SQL interface. The layout is identical to the previous screenshot, with the "RUN SQL" tab selected. The code editor contains the same insert statement as above. The results panel shows the execution details and success message:

insert into hr.staff values
(1,'Steve',rand()\*30000,'Branch\_B'),
(2,'Chris',rand()\*30000,'Branch\_A'),
(3,'Paula',rand()\*30000,'Branch\_A'),
(4,'Craig',rand()\*30000,'Branch\_B'),
(5,'Pete',rand()\*30000,'Branch\_A'),
(6,'Stephanie',rand()\*30000,'Branch\_B'),
(7,'Julie',rand()\*30000,'Branch\_B'),
(8,'Chrissie',rand()\*30000,'Branch\_A');

insert into hr.staff va... Run time: 9.964s  
Status: Success | Affected Rows: 8

You created a staff table and populated it with some data. Now, take a look at the data.

- 14. Run this statement to select from the table.

```
select * from hr.staff
```

The screenshot shows the IBM Db2 Big SQL interface. In the 'RUN SQL' tab, there is a code editor with a single line of SQL: 'select \* from hr.staff;'. Below the code editor is a results panel. The results panel header says 'Result - 09/20/2011 11:48:01'. It shows a table with 8 rows of data:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
1	Steve	13792.16	Branch_B
2	Chris	2221.07	Branch_A
3	Paula	10255.41	Branch_A
4	Craig	10630.80	Branch_B
5	Pete	21869.84	Branch_A
6	Stephanie	5170.30	Branch_B
7	Julie	27180.44	Branch_B
8	Chrissie	1703.34	Branch_A

## Part 2: Creating roles and assigning privileges to the roles

In this part, you create the roles and assign the required privileges to the roles by using the grant command. Then, the role has the wanted privileges on the required tables, and it is ready to be granted to different users on the system. After the role is assigned to a user, this user has the same privileges of the role.

- 1. Copy/paste these statements to create the roles and assign the privileges to them.

```
CREATE ROLE BRANCH_A_ROLE;
GRANT SELECT ON HR.STAFF TO ROLE BRANCH_A_ROLE;
CREATE ROLE BRANCH_B_ROLE;
GRANT SELECT ON HR.STAFF TO ROLE BRANCH_B_ROLE;

CREATE ROLE FINANCE_ROLE;
GRANT SELECT ON HR.STAFF TO ROLE FINANCE_ROLE;
```

- 2. Run the statements.

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for discover, help, and user profile. Below the bar, it says "RUN SQL". A tab labeled "\* Untitled - 1" is open. The SQL code in the editor is:

```

1 CREATE ROLE BRANCH_A_ROLE;
2 GRANT SELECT ON HR.STAFF TO ROLE BRANCH_A_ROLE;
3
4 CREATE ROLE BRANCH_B_ROLE;
5 GRANT SELECT ON HR.STAFF TO ROLE BRANCH_B_ROLE;
6
7 CREATE ROLE FINANCE_ROLE;
8 GRANT SELECT ON HR.STAFF TO ROLE FINANCE_ROLE;
9

```

On the right side, under "Result - 09/20/2011 11:50:08", the output is displayed with run times for each statement:

- > ✓ CREATE ROLE BRANCH\_A\_RO... Run time: 0.031s
- > ✓ GRANT SELECT ON HR.STAF... Run time: 0.017s
- > ✓ CREATE ROLE BRANCH\_B\_RO... Run time: 0.011s
- > ✓ GRANT SELECT ON HR.STAF... Run time: 0.009s
- > ✓ CREATE ROLE FINANCE\_ROLE Run time: 0.007s
- > ✓ GRANT SELECT ON HR.STAF... Run time: 0.020s

### Part 3: Creating and enforcing the permissions

In this part, you create the permissions for the different roles. The CREATE PERMISSION statement is an independent statement that can be used to create a row permission before row access control is activated for a table. The only requirement is that the table and the columns exist before the permission is created. Multiple row permissions can be created for a table.

Allow the role FINANCE\_ROLE to see all rows of data. This role has the highest level of access.

- 1. On the canvas, type or copy and paste the following statements.

```

CREATE PERMISSION FINANCE_ACCESS ON HR.STAFF
FOR ROWS WHERE

VERIFY_ROLE_FOR_USER(SESSION_USER, 'FINANCE_ROLE') = 1

ENFORCED FOR ALL ACCESS
ENABLE;

```

- 2. Run the statements.

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for discover, help, and user profile. Below the bar, it says "RUN SQL". A tab labeled "\* Untitled - 1" is open. The SQL code in the editor is:

```

1 CREATE PERMISSION FINANCE_ACCESS ON HR.STAFF
2 FOR ROWS WHERE
3
4 VERIFY_ROLE_FOR_USER(SESSION_USER, 'FINANCE_ROLE') = 1
5
6 ENFORCED FOR ALL ACCESS
7 ENABLE;
8

```

On the right side, under "Result - 09/20/2011 11:51:51", the output is displayed with a success message:

✓ CREATE PERMISSION FINAN... Run time: 0.122s

Status: Success | Affected Rows: 0

- \_\_\_ 3. Run the following statement to allow the role **BRANCH\_A\_ROLE** to see **Branch\_A** data only:

```
CREATE PERMISSION BRANCH_A_ACCESS ON HR.STAFF
FOR ROWS WHERE
(
    VERIFY_ROLE_FOR_USER(SESSION_USER, 'BRANCH_A_ROLE') = 1
    AND
    HR.STAFF.BRANCH_NAME = 'Branch_A'
)
ENFORCED FOR ALL ACCESS
ENABLE;
```

A permission is created to give access to **Branch\_A** to users in the role **BRANCH\_A\_ROLE**.

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for three horizontal bars, 'IBM Db2 Big SQL', a magnifying glass labeled 'Discover', a question mark, and a user profile. Below the navigation bar, the title 'RUN SQL' is visible. Underneath it, a tab labeled '\* Untitled - 1' is selected. The main area contains the following SQL code:

```
1 CREATE PERMISSION BRANCH_A_ACCESS ON HR.STAFF
2 FOR ROWS WHERE
3 (
4     VERIFY_ROLE_FOR_USER(SESSION_USER, 'BRANCH_A_ROLE') = 1
5     AND
6     HR.STAFF.BRANCH_NAME = 'Branch_A'
7 )
8 ENFORCED FOR ALL ACCESS
9 ENABLE;
```

To the right of the code, the results are displayed in a panel titled 'Result - 09/20/2011 11:52:30'. It shows a single row with a green checkmark icon, the text 'CREATE PERMISSION BRANC...', 'Run time: 0.022s', and a status message 'Status: Success | Affected Rows: 0'.

- \_\_\_ 4. Run the following statement to allow **BRANCH\_B\_ROLE** to see **Branch\_B** data only:

```
CREATE PERMISSION BRANCH_B_ACCESS ON HR.STAFF
FOR ROWS WHERE
(
    VERIFY_ROLE_FOR_USER(SESSION_USER, 'BRANCH_B_ROLE') = 1
    AND
    HR.STAFF.BRANCH_NAME = 'Branch_B'
)
ENFORCED FOR ALL ACCESS
ENABLE;
```

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for 'Discover' and a user profile. Below the navigation bar, the title 'IBM Db2 Big SQL' is displayed. On the left, a 'RUN SQL' tab is selected, showing a script editor with a single line of code: 'CREATE PERMISSION BRANCH\_B\_ACCESS ON HR.STAFF'. The code is highlighted in blue. To the right of the editor is a results panel with the message 'Result - 09/20/2011 11:53:37'. Below the message, it says 'CREATE PERMISSION BRANC...' with a green checkmark and 'Run time: 0.041s'. At the bottom of the results panel, it shows 'Status: Success | Affected Rows: 0'.

## Part 4: Granting roles to users and activating the row access control

In this part, you grant the different roles to the users. After the role is granted, each user has the privileges of the corresponding role.

- 1. Grant the user `branch_a` the role `BRANCH_A_ROLE`. Grant the user `branch_b` the role `BRANCH_B_ROLE`. Grant the user `cfo` the role `FINANCE_ROLE`.

On the canvas, type or copy/paste the following statements.

```
GRANT ROLE BRANCH_A_ROLE TO USER Branch_a;
GRANT ROLE BRANCH_B_ROLE TO USER Branch_b;
GRANT ROLE FINANCE_ROLE TO USER Cfo;
```

- 2. Run the statements.

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for 'Discover' and a user profile. Below the navigation bar, the title 'IBM Db2 Big SQL' is displayed. On the left, a 'RUN SQL' tab is selected, showing a script editor with three lines of code: 'GRANT ROLE BRANCH\_A\_ROLE TO USER Branch\_a;', 'GRANT ROLE BRANCH\_B\_ROLE TO USER Branch\_b;', and 'GRANT ROLE FINANCE\_ROLE TO USER Cfo;'. The code is highlighted in blue. To the right of the editor is a results panel with the message 'Result - 01/20/21 11:54:22'. Below the message, it shows three separate grants: 'GRANT ROLE BRANCH\_A\_ROLE TO USER Branch\_a' (Run time: 0.010s), 'GRANT ROLE BRANCH\_B\_ROLE TO USER Branch\_b' (Run time: 0.004s), and 'GRANT ROLE FINANCE\_ROLE TO USER Cfo' (Run time: 0.000s). Each grant has a green checkmark and the status 'Success | Affected Rows: 0'.

- 3. Activate the row access control.

On the canvas, type or copy and paste the following statements.

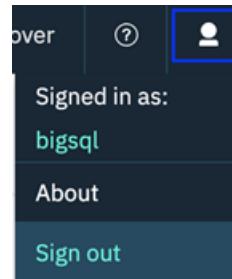
```
ALTER TABLE HR.STAFF
ACTIVATE ROW ACCESS CONTROL;
```

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for 'Discover' and a user profile. Below the navigation bar, the title 'IBM Db2 Big SQL' is displayed. On the left, a 'RUN SQL' tab is selected, showing a script editor with two lines of code: 'ALTER TABLE HR.STAFF' and 'ACTIVATE ROW ACCESS CONTROL;'. The code is highlighted in blue. To the right of the editor is a results panel with the message 'Result - 09/20/2011 11:55:58'. Below the message, it shows the command 'ALTER TABLE HR.STAFF AC...' with a green checkmark and 'Run time: 0.130s'. At the bottom of the results panel, it shows 'Status: Success | Affected Rows: 0'.

## **Part 5: Switching between users in Db2 Big SQL Console**

To go through the rest of this demonstration, you switch between the three users that you set up to demonstrate row and column access control.

- \_\_\_ 1. At the upper right, click the avatar icon and select **Sign out**.



- \_\_\_ 2. Sign in as user **branch\_a**.

## **Part 6: Using row access control**

In this part, you run a query by using different users with different roles, and inspect the results that are returned for each user according to the user role.

- \_\_\_ 1. As the user **branch\_a**, click **Run SQL** from the left navigation and choose **Blank**.

```
SELECT * FROM HR.STAFF;
```

The screenshot shows the IBM Db2 Big SQL interface. In the left panel, under 'RUN SQL', there is a code editor with the query: 'SELECT \* FROM HR.STAFF'. In the right panel, the results of the query are displayed in a table. The table has columns: EMPNO, FIRST\_NAME, SALARY, and BRANCH\_NAME. The data returned is for employees in Branch\_A:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
2	Chris	2221.07	Branch_A
3	Paula	10255.41	Branch_A
5	Pete	21869.84	Branch_A
8	Chrissie	1703.34	Branch_A

Only the rows from **Branch\_A** are returned because user **branch\_a** is part of the role that has access only to **Branch\_A**

- \_\_\_ 2. Sign out and log in as the user **branch\_b**.
- \_\_\_ 3. Click **Run SQL** from the left navigation and choose **Blank**.
- \_\_\_ 4. Select rows from the HR.STAFF table.

```
SELECT * FROM HR.STAFF;
```

The screenshot shows the IBM Db2 Big SQL interface. In the left panel, under 'RUN SQL', there is a code editor with the query: 'SELECT \* FROM HR.STAFF;'. In the right panel, the results of the query are displayed in a table. The table has columns: EMPNO, FIRST\_NAME, SALARY, and BRANCH\_NAME. The data returned is for employees in Branch\_B:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
1	Steve	13792.16	Branch_B
4	Craig	10630.80	Branch_B
6	Stephanie	5170.30	Branch_B
7	Julie	27180.44	Branch_B

Likewise, user **branch\_b** has access only to **Branch\_B** through this role.

- \_\_\_ 5. Sign out and log in as the user **cfo**.
- \_\_\_ 6. Click **Run SQL** from the left navigation and choose **Blank**.
- \_\_\_ 7. Select rows from the **HR.STAFF** table.

```
SELECT * FROM HR.STAFF;
```

The screenshot shows the IBM Db2 Big SQL interface. In the top navigation bar, there are icons for discover, help, and user profile. Below the bar, it says "RUN SQL". A tab labeled "\* Untitled - 1" is selected. The main area contains a toolbar with various icons like copy, paste, search, and syntax assistant. Below the toolbar, a query is written: "1 SELECT \* FROM HR.STAFF;". To the right, the results are displayed in a table format. The table has four columns: EMPNO, FIRST\_NAME, SALARY, and BRANCH\_NAME. The data consists of 8 rows:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
1	Steve	13792.16	Branch_B
2	Chris	2221.07	Branch_A
3	Paula	10255.41	Branch_A
4	Craig	10630.80	Branch_B
5	Pete	21869.84	Branch_A
6	Stephanie	5170.30	Branch_B
7	Julie	27180.44	Branch_B
8	Chrissie	1703.34	Branch_A

At the bottom right of the results pane, there is a link "Show Less".

The user `CFO` is part of the `FINANCE` role, so CFO has access to all the data.

## Part 7: Shielding data with column masking

The CREATE MASK statement is an independent statement that can be used to create a column access control mask before column access control is activated for a table. The only requirement is that the table and the columns exist before the mask is created. Multiple column masks can be created for a table but a column can have one mask only.

In the cases where the users `branch_a` and `branch_b` select data from the `HR.STAFF` table, they are able to see the salary information. Salary is sensitive information so it must be hidden from everyone except from users in the `FINANCE` role.

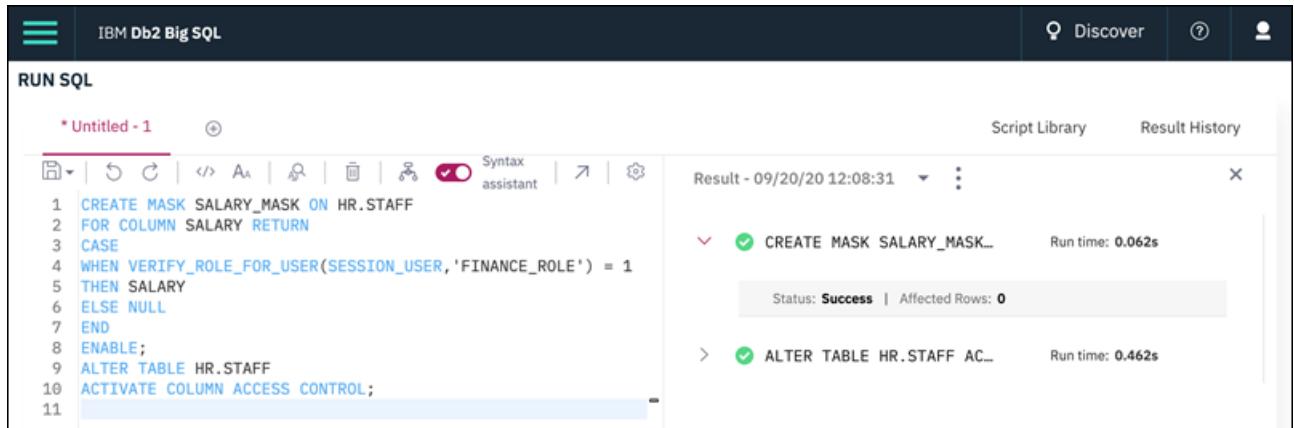
- \_\_\_ 1. Sig out and log in as the user `bigsq1`.
- \_\_\_ 2. Click **RUN SQL** from the left navigation menu, then click **Blank**.
- \_\_\_ 3. Paste the following statements then click **Run all**:

```

CREATE MASK SALARY_MASK ON HR.STAFF
FOR COLUMN SALARY RETURN
CASE
WHEN VERIFY_ROLE_FOR_USER(SESSION_USER, 'FINANCE_ROLE') = 1
THEN SALARY
ELSE NULL
END
ENABLE;
ALTER TABLE HR.STAFF
ACTIVATE COLUMN ACCESS CONTROL;

```

These statements create a mask on the SALARY column where if users are part of the FINANCE\_ROLE, they see the salary information. Otherwise, they get **NULL**.



The screenshot shows the IBM Db2 Big SQL interface. In the 'RUN SQL' tab, a script named '\* Untitled - 1' is being run. The code creates a column mask for the 'SALARY' column in the 'HR.STAFF' table. The mask returns the salary if the user is in the 'FINANCE\_ROLE'; otherwise, it returns NULL. It also activates column access control. The results pane shows two successful operations: 'CREATE MASK SALARY\_MASK' and 'ALTER TABLE HR.STAFF AC...'. Both operations took less than a second.

```

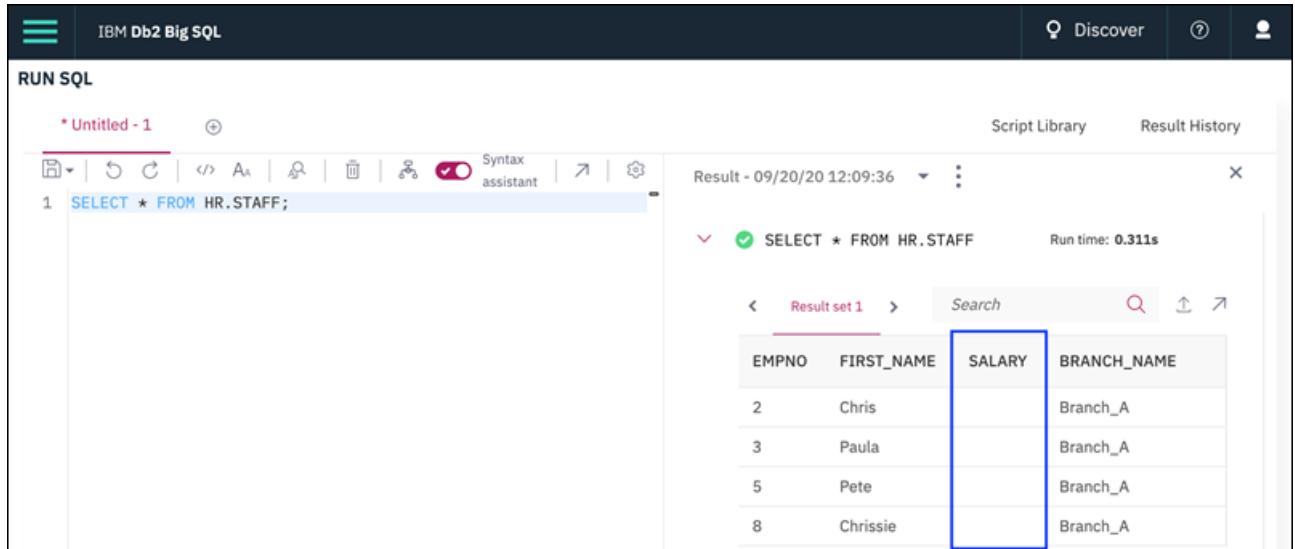
1 CREATE MASK SALARY_MASK ON HR.STAFF
2 FOR COLUMN SALARY RETURN
3 CASE
4 WHEN VERIFY_ROLE_FOR_USER(SESSION_USER, 'FINANCE_ROLE') = 1
5 THEN SALARY
6 ELSE NULL
7 END
8 ENABLE;
9 ALTER TABLE HR.STAFF
10 ACTIVATE COLUMN ACCESS CONTROL;
11

```

- 4. Switch to each of the three users and run the select statement to select from the HR.STAFF table.

```
SELECT * FROM HR.STAFF;
```

As user **branch\_a**:



The screenshot shows the IBM Db2 Big SQL interface. A user named 'branch\_a' is connected. In the 'RUN SQL' tab, a single line of code 'SELECT \* FROM HR.STAFF;' is run. The results pane shows the output of the query, which includes columns: EMPNO, FIRST\_NAME, SALARY, and BRANCH\_NAME. The 'SALARY' column is highlighted with a blue border, indicating that it contains NULL values for users not in the 'FINANCE\_ROLE'.

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
2	Chris		Branch_A
3	Paula		Branch_A
5	Pete		Branch_A
8	Chrissie		Branch_A

As user **branch\_b**:

The screenshot shows the IBM Db2 Big SQL interface. In the 'RUN SQL' tab, a query is run:

```
1 SELECT * FROM HR.STAFF;
```

The results are displayed in a table:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
1	Steve		Branch_B
4	Craig		Branch_B
6	Stephanie		Branch_B
7	Julie		Branch_B

As user **cfo**:

The screenshot shows the IBM Db2 Big SQL interface. In the 'RUN SQL' tab, a query is run:

```
1 SELECT * FROM HR.STAFF;
```

The results are displayed in a table:

EMPNO	FIRST_NAME	SALARY	BRANCH_NAME
1	Steve	13792.16	Branch_B
2	Chris	2221.07	Branch_A
3	Paula	10255.41	Branch_A
4	Craig	10630.80	Branch_B
5	Pete	21869.84	Branch_A
6	Stephanie	5170.30	Branch_B
7	Julie	27180.44	Branch_B
8	Chrissie	1703.34	Branch_A

**End of exercise**

## Exercise review and wrap-up

In this exercise, you created row and column-based access control to protect sensitive data through dynamic filtering (row level security) and dynamic masking (column level security).

# Exercise 5. Configuring impersonation in Db2 Big SQL



## Important

This exercise requires administration authority that is not granted to the students in this course. This exercise can be performed only by the instructor. A recorded demo of this exercise is available.

## Estimated time

00:45

## Overview

Impersonation is the ability to allow a service user to securely access data in Hadoop on behalf of another user. If you enable impersonation at the global level in Db2 Big SQL, the `bigsq1` user can impersonate the connected user to perform actions on Hadoop tables. For example, if the service user needs to access a set of data that belongs to user1, impersonation allows `bigsq1` to access and perform actions on the data that belongs to user1. Also, when you issue CREATE HADOOP TABLE, run a query, or load an operation, Db2 Big SQL performs the operations in HDFS and Hive as the connected user.

Note that by default, the `bigsq1` user performs the read and write operations on HDFS, Hive, and HBase that are required for the Db2 Big SQL service. You do not need impersonation if the `bigsq1` user is the sole owner and the user of the data.

In this exercise, you enable and configure impersonation in Db2 Big SQL.

For more information on Db2 Big SQL impersonation, see Impersonation in Db2 Big SQL at:

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_7.1.0/com.ibm.swg.im.bigsq1.doc/doc/big\\_sq1\\_imperonate.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_7.1.0/com.ibm.swg.im.bigsq1.doc/doc/big_sq1_imperonate.html)

## Objectives

After completing this exercise, you will be able to:

- Configure impersonation in Db2 Big SQL.
- Set up impersonation for non-admin users so that they can create a Hadoop table, load data into it, and run queries.

## Requirements

- PuTTY SSH client installed in your workstation.
- For information about the variables used in this exercise, refer to the table in Exercise 1. “Connecting to the IBM Db2 Big SQL server”.

## Exercise instructions

In this exercise, you complete the following tasks:

- \_\_\_ 1. Enable impersonation in Db2 Big SQL.
- \_\_\_ 2. Set the permissions to the "warehouse" directory on HDFS.
- \_\_\_ 3. Copy the sample data to HDFS.
- \_\_\_ 4. Create a Hadoop table, load data, and query the table.

### Part 1: Enabling impersonation in Db2 Big SQL

In this part, you enable impersonation in Db2 Big SQL.

Confirm that the **HDFS** settings required for Db2 Big SQL impersonation are set by completing the following steps:

- \_\_\_ 1. From the Ambari Web UI, select the **HDFS** service from the left navigation.
- \_\_\_ 2. Select the **CONFIGS** tab.
- \_\_\_ 3. Click the **ADVANCED** tab.
- \_\_\_ 4. Type the property name in the Filter bar  
**hadoop.proxyuser.bigsq1**

- \_\_\_ 5. Verify that the properties are set.

If not, add them and set them as shown in the figure. These properties should be configured automatically on installation, but it is a good practice to check this prerequisite.

Next, check the authorization settings in Hive.

- \_\_\_ 6. Click the **Hive** service on the navigation.
- \_\_\_ 7. Select the **CONFIGS** tab.
- \_\_\_ 8. Ensure that **Hive** is running with **Authorization = None**.
- \_\_\_ 9. Ensure that **Run as end user instead of Hive user = True**.

The screenshot shows the 'CONFIGS' tab selected in the top navigation bar. Under the 'SETTINGS' tab, there are two main sections: 'Interactive Query' and 'Security'. The 'Interactive Query' section has a 'No' button and edit/copy icons. The 'Security' section includes 'Choose Authorization' (set to 'None'), 'Run as end user instead of Hive user' (set to 'True'), and 'HiveServer2 Authentication' (set to 'None'). Each section has edit and add icons.

Verify that *inherit permissions* in Hive is disabled. This property impacts the default permissions granted to new directories under `/warehouse`.

- \_\_\_ 10. Click the **ADVANCED** tab.
- \_\_\_ 11. Expand the **Custom hive-site** section.
- \_\_\_ 12. If the property `hive.warehouse.subdir.inherit.perms` does not exist, then click **Add Property**, and enter the following values then click **ADD**.
  - Key: **hive.warehouse.subdir.inherit.perms**
  - Value: **false**

Add Property

Type	hive-site.xml		
Key	hive.warehouse.subdir.inherit.perms		
Value	false		
Property Type	PASSWORD USER GROUP TEXT		
<input type="button" value="CANCEL"/> <input style="background-color: green; color: white; border: none; font-weight: bold;" type="button" value="ADD"/>			

- 13. If the property exists, then verify that its value is **false**.

Ambari

Dashboard Services

- HDFS
- YARN
- MapReduce2
- Tez
- Hive**
- HBase
- Pig
- Sqoop
- Oozie

Custom hive-site

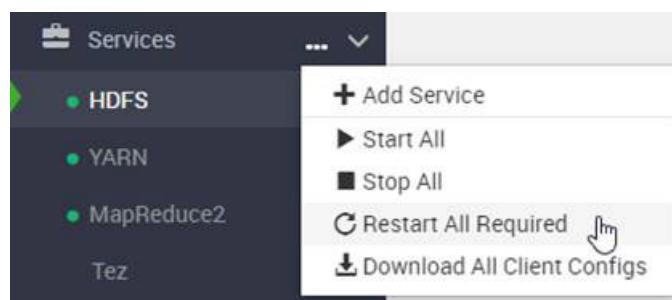
atlas.rest.address	http://dataengineer.ibm.com:21000
bigsqll.catalog.sync.events	/user/bigsqll/sync
bigsqll.catalog.sync.exclude	
bigsqll.catalog.sync.include	
bigsqll.catalog.sync.sleep	30
fs.file.impl.disable.cache	true
fs.hdfs.impl.disable.cache	true
<b>hive.warehouse.subdir.inherit.perms</b>	<b>false</b>

- 14. Click **Save** to save the changes and accept all recommended configurations.

Now, check the UMASK value in the Ambari Web UI.

- 15. Select the **HDFS** service.
- 16. Select the **CONFIGS** tab.
- 17. Click the **ADVANCED** tab.
- 18. Enter **umask** in the **Filter** field.
- 19. Verify that the default value is **022** as shown in the following figure.

- \_\_\_ 20. Restart all services that need a restart by clicking the Services actions menu, then clicking Restart All Required.



## **Part 2: Setting the permissions to the warehouse directory on HDFS**

In this part, you set the permissions for the HDFS directory **warehouse** to 777 so that all users can write to the directory. For example, when you create a schema, a directory is created under the **warehouse** folder. To allow users to create their own schemas, you need to set the inherit permissions properties to “false” and set the **warehouse** directory permissions to 777.

- \_\_\_ 1. Open a terminal or use an existing one.  
\_\_\_ 2. As the **root** user, create a new user, **henry**, and set the password for the new user.

```
useradd -g hdfs henry  
passwd henry (Enter the password and retype the new password)
```

- \_\_\_ 3. Switch to the **hdfs** user:
- ```
su - hdfs
```
- \_\_\_ 4. Set the permissions on the **warehouse** directory in HDFS to **777**.
- ```
hdfs dfs -chmod 777 /warehouse
```

- \_\_\_ 5. Create a folder in HDFS for the user **henry** and set the appropriate permissions.

```
hdfs dfs -mkdir /user/henry  
hdfs dfs -chown henry:henry /user/henry  
hdfs dfs -chmod 755 /user/henry
```

- \_\_\_ 6. Verify that the folder was properly created with the appropriate permissions.
- ```
hdfs dfs -ls /user | grep henry
```

```
[hdfs@dataengineer ~]$ hdfs dfs -ls /user | grep henry
drwxr-xr-x  - henry      henry          0 2021-01-15 04:29 /user/henry
```

Keep this terminal open. You will use it in the next task.

### **Part 3: Copying the sample data to HDFS**

In this part, you copy the sample data file from the local file system, into the user's hdfs file system to prepare it for loading into the Hadoop table in the next part.

Complete these steps:

\_\_\_ 1. SSH into your server by using PuTTY or similar tool and log in as user **henry**.

\_\_\_ 2. Create a data directory.

```
hdfs dfs -mkdir /user/henry/data
```

\_\_\_ 3. Copy the SLS\_SALES\_FACT.txt file into the data directory.

```
hdfs dfs -put
/usr/ibmpacks/current/bigsql/bigsql/samples/data/GOSALESDW.SLS_SALES_FACT.txt
/t /user/henry/data/
```

\_\_\_ 4. Verify that the copy was successful.

```
hdfs dfs -ls /user/henry/data
```

The results are similar to the following output.

```
[henry@dataengineer ~]$ hdfs dfs -ls /user/henry/data
Found 1 items
-rw-r--r--  3 henry henry  51258157 2020-10-26 15:54 /user/henry/data/GOSALESDW.SLS_SALES_FACT.txt
```

### **Part 4: Creating a Hadoop table, loading data, and querying the table**

In this part, you create a Hadoop table and load it with the data from the sample data file that you copied in the previous part.

Complete these steps:

- \_\_\_ 1. Log in to the Ambari Web UI by entering the Ambari URL <hostname:8080> and log in as Ambari admin <ambari admin>.
- \_\_\_ 2. Click **Big SQL Console > Console UI**.
- \_\_\_ 3. Log in as user **henry**.
- \_\_\_ 4. From the hamburger menu, click **RUN SQL** then click **Blank**.

- \_\_\_ 5. Paste the following create statement to create the fact table **sls\_sales\_fact**.

```
CREATE HADOOP TABLE IF NOT EXISTS sls_sales_fact
( order_day_key INT NOT NULL
, organization_key INT NOT NULL
, employee_key INT NOT NULL
, retailer_key INT NOT NULL
, retailer_site_key INT NOT NULL
, product_key INT NOT NULL
, promotion_key INT NOT NULL
, order_method_key INT NOT NULL
, sales_order_key INT NOT NULL
, ship_day_key INT NOT NULL
, close_day_key INT NOT NULL
, quantity INT
, unit_cost DOUBLE
, unit_price DOUBLE
, unit_sale_price DOUBLE
, gross_margin DOUBLE
, sale_total DOUBLE
, gross_profit DOUBLE)

ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
```

- \_\_\_ 6. Run the statement.

The screenshot shows the IBM Db2 Big SQL interface. At the top, there are navigation icons for 'Discover' (with a magnifying glass), '?', and a user profile icon. Below the header, there are two tabs: 'Script Library' and 'Result History'. The main area displays the execution results for the 'CREATE TABLE' command. The output shows a green checkmark indicating success, followed by the executed SQL statement and its run time (9.287s). A status bar at the bottom shows 'Status: Success | Affected Rows: 0'.

Next, load the sample data into the table.

- \_\_\_ 7. Paste the following **load** statement. This statement loads the data that you placed on the HDFS earlier.

```
load hadoop using file url '/user/henry/data/GOSALESdw.SLS_SALES_FACT.txt'
with SOURCE PROPERTIES ('field.delimiter'='\t') INTO TABLE SLS_SALES_FACT
overwrite;
```

- \_\_\_ 8. Run the statement.

The screenshot shows a successful Hadoop load operation. The command run was: `load hadoop using file url '/user/henry/data/GOSALESdw.SLS_SALES_FACT.t...`. The status is **Success** and affected rows are 0. The run time was 47.633s.

9. Verify that the table was created and that you can select from it.

On the Run SQL canvas, still as user **henry**, run the following **select all** statement on the **sls\_sales\_fact** table.

```
select * from sls_sales_fact;
```

The screenshot shows the results of a `select * from sls_sales_fact` query. The results are displayed in a table titled "Result set 1".

| ORDER_DAY_KEY | ORGANIZATION_KEY | EMPLOYEE_KEY | RETAILER_KEY | RETAILER_SITE_KEY | PRODUCT_KEY |
|---------------|------------------|--------------|--------------|-------------------|-------------|
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30025       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30026       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30030       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30032       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30035       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30039       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30063       |
| 20040112      | 11104            | 4072         | 6748         | 5038              | 30066       |

The **bigsq1** user impersonates the user **henry** and runs queries on behalf of this user. Without impersonation, the user **henry** would need to be granted access to the database, and tables. With one user, this approach is manageable, but imagine a scenario where there are several users with multiple database objects. Such scenario would require many access controls.

Think about what would happen if the user `henry` tries to create a different table without impersonation enabled. You will see that `henry` would not be able to create this table due to insufficient privileges.

```
=====DB2 LOG=====
```

```
The statement failed because the authorization ID does not have the required authorization or  
privilege to perform the operation. Authorization ID: "BIGSQL". Operation: "Hive access control error  
[BSL-0-716f59af8]". Object: "UNKNOWN".. SQLCODE=-551, SQLSTATE= 4250, DRIVER=3.72.16  
Learn more about this error
```

## End of exercise

## Exercise review and wrap-up

You set up impersonation in Db2 Big SQL so that a non-admin user, such as *henry*, is able to load and run queries on the Hadoop tables by using Db2 Big SQL without having to be granted explicit privileges.

# Exercise 6. Getting started with Db2 Big SQL federation



## Important

Instructors must complete the steps described in "[Steps to run before Big SQL Exercise 6](#)" before the students can perform Part 2 of this exercise.

## Estimated time

01:00

## Overview

Within a federated system, a single SQL statement can access data that is distributed among one or more data sources. The federation that is bundled in IBM® Db2® Big SQL supports various data sources.

In this exercise, you follow the basic steps to get started with the federation feature of Db2 Big SQL.

For more information on Db2 Big SQL federation, see Federation at

[https://www.ibm.com/support/knowledgecenter/SSCRJT\\_6.0.0/com.ibm.swg.im.bigsqI.federation.doc/federation\\_overview.html](https://www.ibm.com/support/knowledgecenter/SSCRJT_6.0.0/com.ibm.swg.im.bigsqI.federation.doc/federation_overview.html)

## Objectives

After completing this exercise, you will be able to:

- Create a Db2 server instance on IBM Cloud.
- Create a server object to connect your federated Db2 Big SQL server to the remote data source.
- Create user mapping objects.
- Create a nickname.
- Run queries from the federated Db2 Big SQL server to query the data on the Db2 service instance on IBM Cloud.

## Introduction

With Db2 Big SQL, you can efficiently query data on Hadoop and also combine data that is spread around different enterprise data warehouses. The federation capability of Db2 Big SQL lets you query against and combine with Hadoop data.

Federation enables you to send distributed requests to multiple data sources within a single SQL statement.

In a federated database, a query can access data that is stored on any of the data sources that comprise the federated database.

Complete the following steps to set up the federation:

1. Set up the federation environment.

This step is not required for all data sources. It can include the installation of a client for your intended data source. It usually involves setting some environmental variables in the `db2dj.ini` file under your `sql1ib/cfg` directory.

2. Create a server object.

The server object maps to exactly one database. The CREATE SERVER statement requires the definition of a server type. For more information, see the [CREATE SERVER statement](#).

3. Create user mapping objects.

These objects map local authorizations to remote authorizations. They are not specific to a particular type of data source. For more information, see the [CREATE USER MAPPING statement](#).

4. Create a nickname.

This step is not necessarily required to access data. For example, with Db2, it is possible to access data after you create a user mapping by referencing a three-part name (`SERVER.REMOTE_USER.REMOTE_TABLE`) without creating a nickname. There are advantages to using nicknames. The most relevant advantage is enhanced performance by using statistics. For more information, see the [CREATE NICKNAME statement](#).

In this exercise, you perform most of these basic steps. For more information, see [Getting started with federation](#).

## Requirements

- IBM Cloud account
- For information about the variables used in this exercises, refer to the table in Exercise 1. “Connecting to the IBM Db2 Big SQL server”.



### Note

The hostnames that are shown in the screen captures of this exercise might be different from the one in your hands-on environment.

# Exercise instructions

In this exercise, you complete the following tasks:

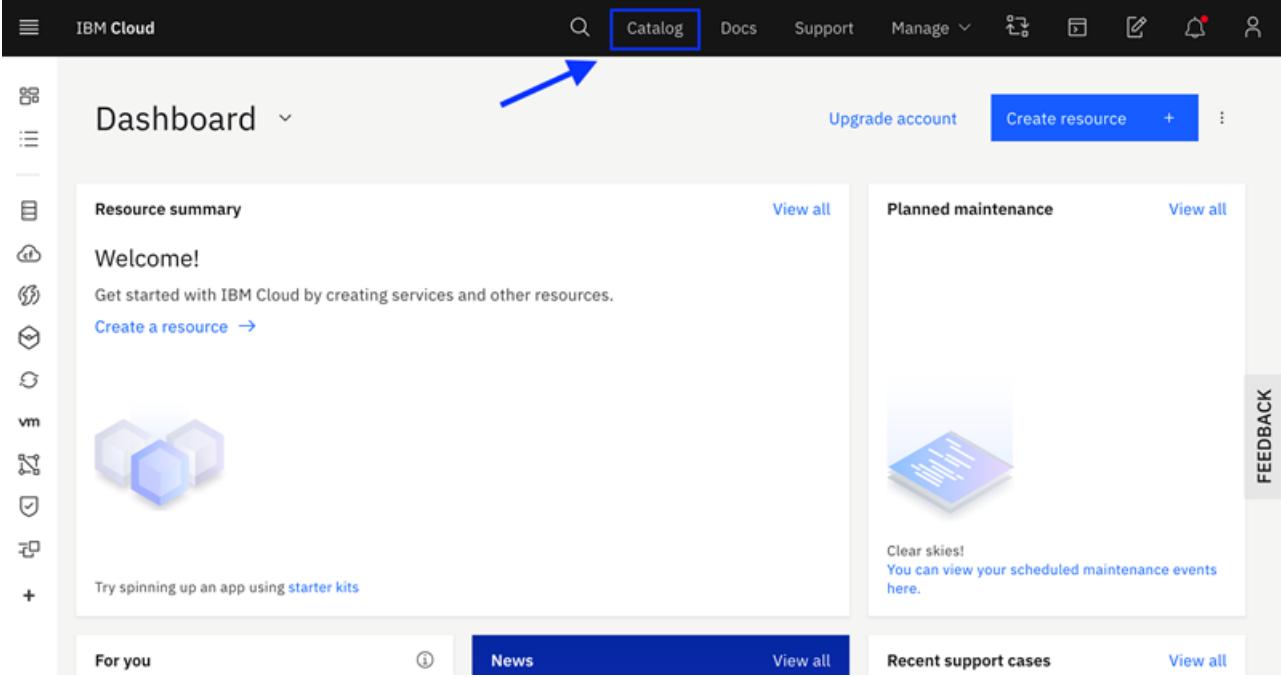
- \_\_\_ 1. Prepare the remote data source for this exercise.
- \_\_\_ 2. Create a server object.
- \_\_\_ 3. Create the user mapping object.
- \_\_\_ 4. Create the nickname.
- \_\_\_ 5. Run queries.

## **Part 1: Preparing the remote data source for this exercise**

In this part, you prepare your Relational Database Management System (RDBMS) for the Db2 Big SQL federation. This exercise uses Db2 on Cloud. In this part, you create an instance of the Db2 service on IBM Cloud. You also create a table on the Db2 database and load data to the table.

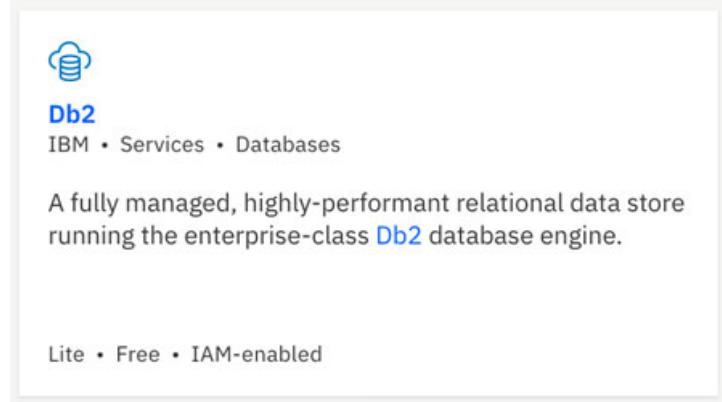
Complete the following steps to create an instance of the Db2 service on IBM Cloud:

- \_\_\_ 1. Log in to IBM Cloud with your IBM Cloud account.  
<https://cloud.ibm.com/login/>
- \_\_\_ 2. Click **Catalog**.



The screenshot shows the IBM Cloud dashboard. At the top, there's a navigation bar with 'IBM Cloud' and various links like 'Docs', 'Support', 'Manage', and a search bar. A blue arrow points to the 'Catalog' button, which is highlighted with a blue border. Below the navigation bar, there's a 'Dashboard' section with a 'Resource summary' card showing 'Welcome!' and a 'Planned maintenance' card. On the left, there's a sidebar with icons for different services like 'vm', 'starter kits', and 'FEEDBACK'. At the bottom, there are sections for 'For you', 'News', and 'Recent support cases'.

- \_\_\_ 3. In the filter field, enter **Db2**.
- \_\_\_ 4. Click the **Db2** service.



- \_\_\_ 5. Select the **Lite** plan.
- \_\_\_ 6. Click **Create**.

The screenshot shows the IBM Cloud service catalog. In the top right, there's a 'Summary' section for a 'Db2' service in 'London' with a 'Plan: Lite' and a 'Free' status. On the left, a 'Select a pricing plan' dropdown is set to 'London'. Below it, a table compares 'Plan', 'Features', and 'Pricing' for 'Lite' and 'Standard' plans. The 'Lite' plan is highlighted with a blue box and arrow. The 'Pricing' row for 'Lite' shows 'Free'. The 'Standard' plan details include 'Instance with flexible scaling of compute and storage', 'Base instance starts at 8 GB RAM x 20 GB Storage', and various hourly rates. A large blue 'Create' button at the bottom is also highlighted with a blue box and arrow.

- \_\_\_ 7. Click **Service Credentials** as shown in the following figure.

Resource list /

**Db2-ev** Active Add tags

Getting started Manage **Service credentials**

Connections

**Service credentials**

You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud™ service. [Learn more](#)

Search credentials... New credential +

| <input type="checkbox"/> Key name | Date created |
|-----------------------------------|--------------|
|-----------------------------------|--------------|

BACK

\_\_\_ 8. Click **New Credentials**.

\_\_\_ 9. In the Create credential dialog, enter **admin** in the Name field, and click **Add**.

Create credential

Name:

admin

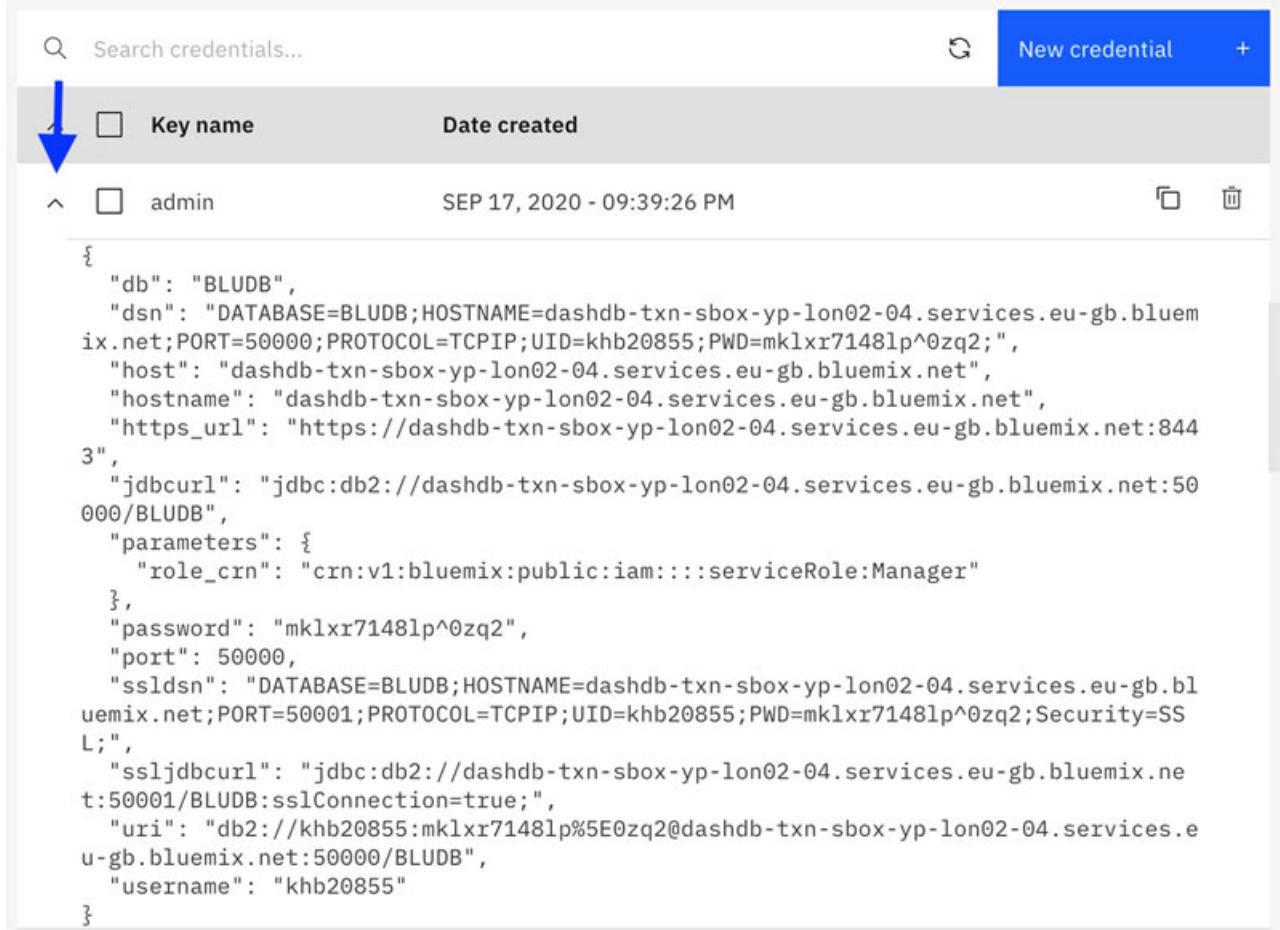
Role:

Manager

Advanced options

Cancel Add

\_\_\_ 10. When the credential is created, expand credentials by clicking the small arrow beside the **admin** credential.



A screenshot of a web-based interface for managing service credentials. At the top, there's a search bar labeled "Search credentials..." and a blue button labeled "New credential". Below the search bar is a table with two columns: "Key name" and "Date created". A single row is visible, showing "admin" as the key name and "SEP 17, 2020 - 09:39:26 PM" as the date created. To the right of this row are icons for deleting and editing. A large blue arrow points down to the expanded JSON content of the "admin" credential. The JSON content is as follows:

```
{
  "db": "BLUDB",
  "dsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net;PORT=50000;PROTOCOL=TCPIP;UID=khb20855;PWD=mklxr7148lp^0zq2;",
  "host": "dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net",
  "hostname": "dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net",
  "https_url": "https://dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net:8443",
  "jdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net:50000/BLUDB",
  "parameters": {
    "role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Manager"
  },
  "password": "mklxr7148lp^0zq2",
  "port": 50000,
  "ssldsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net;PORT=50001;PROTOCOL=TCPIP;UID=khb20855;PWD=mklxr7148lp^0zq2;SecuritySSL;",
  "ssljdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net:50001/BLUDB:sslConnection=true;",
  "uri": "db2://khb20855:mklxr7148lp%5E0zq2@dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net:50000/BLUDB",
  "username": "khb20855"
}
```

11. Click the **Copy to clipboard** icon to copy the credentials and paste them in a text editor  
Make a note of the following values:

- port (this exercise refers to this value as <db2-port>)
- db (this exercise refers to this value as <db2-db>)
- hostname (this exercise refers to this value as <db2-hostname>)
- username (this exercise refers to this value as <db2-username>)
- password (this exercise refers to this value as <db2-password>)



### Note

These values are from your Db2 on Cloud service credentials *not to be confused* with the values for this course that were provided by your instructor in the table in “Exercise 1. Connecting to the IBM Db2 Big SQL server”.

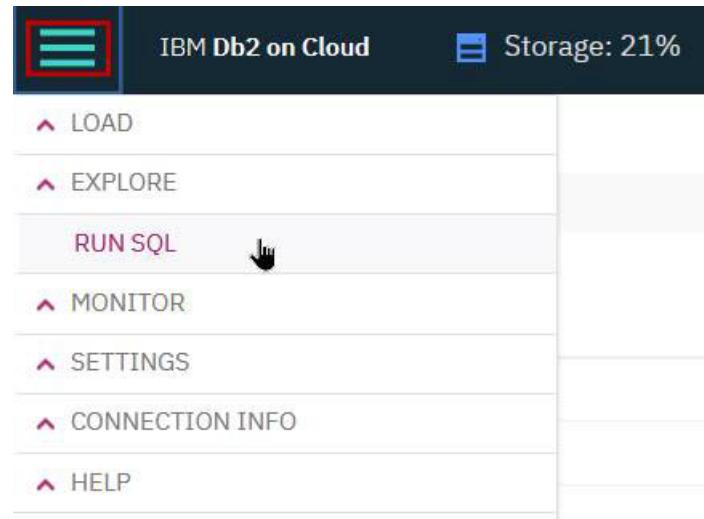
- 
12. On the left, click **Manage**.  
13. Click **Open Console**.

The screenshot shows the IBM Cloud Resource list interface. At the top, it displays 'Resource list /' and the service name 'Db2-oe' with an 'Active' status and an 'Add tags' button. Below this, there's a 'Getting started' section with a 'Manage' button highlighted with a blue border. Other options include 'Service credentials' and 'Connections'. To the right, there's a large blue 'Open Console' button with a cursor icon pointing at it.

- 14. The **IBM Db2 on Cloud Console** opens in a new tab. Does the console look familiar?

The screenshot shows the IBM Db2 on Cloud console interface. At the top, there's a header bar with the IBM logo, the text 'IBM Db2 on Cloud', and a storage status 'Storage: 21%'. Below the header, the word 'TABLES' is prominently displayed. A search bar at the top right contains the placeholder text 'Filter by schema name or table name' and a magnifying glass icon. On the left, there's a 'Schemas' section with a 'Select All' checkbox and a list of schemas: AUDIT, DB2INST1, ERRORSHEMA, GXQ82127, and ST\_INFORMTN\_SCHEMA, each with a corresponding checkbox and a note '(0 table)'.

- 15. From the menu, click **RUN SQL**.



- 16. Select a **Blank** script. Run this statement to create the table **mrk\_promotion\_fact**.
- fact table for marketing promotions

```
CREATE TABLE mrk_promotion_fact
( organization_key INT NOT NULL
, order_day_key INT NOT NULL
, rtl_country_key INT NOT NULL
, employee_key INT NOT NULL
, retailer_key INT NOT NULL
, product_key INT NOT NULL
, promotion_key INT NOT NULL
, sales_order_key INT NOT NULL
, quantity SMALLINT
, unit_cost DOUBLE
, unit_price DOUBLE
, unit_sale_price DOUBLE
, gross_margin DOUBLE
, sale_total DOUBLE
, gross_profit DOUBLE
);
```

The expected output is shown in the following figure.

The screenshot shows the IBM Db2 on Cloud SQL editor. On the left, the SQL code for creating the 'mrk\_promotion\_fact' table is displayed. On the right, the 'Result' pane shows the execution details: 'Result - Oct 4, 2020 2:39...', 'Run time: 0.071s', 'Status: Success | Affected Rows: 0', and a green checkmark icon indicating success.

```
1 CREATE TABLE mrk_promotion_fact
2 { organization_key INT NOT NULL
3 , order_day_key INT NOT NULL
4 , rtl_country_key INT NOT NULL
5 , employee_key INT NOT NULL
6 , retailer_key INT NOT NULL
7 , product_key INT NOT NULL
8 , promotion_key INT NOT NULL
9 , sales_order_key INT NOT NULL
10 , quantity SMALLINT
11 , unit_cost DOUBLE
12 , unit_price DOUBLE
13 , unit_sale_price DOUBLE
14 , gross_margin DOUBLE
15 , sale_total DOUBLE
16 , gross_profit DOUBLE
17 );
```

- 17. Open a terminal in your local workstation and download the file GOSALES DW.MRK\_PROMOTION\_FACT.csv to your local workstation. Run this command and provide your <password> when you are prompted.

```
scp
<username>@<hostname>:/home/bigsq1/labfiles/bigsq1/GOSALES DW.MRK_PROMOTION_F
ACT.csv .
```



### Note

You are prompted to enter your <password> when you run this command.

The expected result is similar to the following output.

```
GOSALES DW.MRK_PROMOTION_FACT.csv 100% 1007KB 367.6KB/s 00:02
```

You use this file to populate the table `mrk_promotion_fact` that you created in the Db2 on Cloud database.

- 18. Go back to the **Db2 on Cloud console**.  
 — 19. From the menu (hamburger icon), select **LOAD -> Load Data**  
 — 20. Select **My Computer** and click **browse files**.

- 21. Select the file **GOSALES DW.MRK\_PROMOTION\_FACT.csv** from your local computer and click **Next**.

## LOAD DATA

You are loading the file **GOSALES DW.MRK\_PROMOTION\_FACT.csv**

**File selection**

Selected file

GOSALES DW.MRK\_PROMOTION\_FACT.csv

My Computer  
A single delimited text file (CSV).  
S3 Amazon S3  
Cloud Object Storage  
Netezza and large CSV file migrations  
Lift

Drag a file here or [browse files](#)

High-speed loads powered by Aspera

Next

- \_\_\_ 22. In the **Select a load target** page, select the <db2-username> value as the **Schema**.
- \_\_\_ 23. Select the table **MRK\_PROMOTION\_FACT**.
- \_\_\_ 24. Select **Overwrite table with new data** in the **Table definition**.
- \_\_\_ 25. Click **Next**

## Select a load target

Refresh

**Schema** New Schema

Find a schema

AUDIT

DB2INST1

ERRORSCHEMA Sample

KHB20855

ST\_INFORMTN\_SCHEMA Sample

**Table** New Table

Find a table in KHB20855

MRK\_PROMOTION\_FACT

**Table definition**

MRK\_PROMOTION\_FACT Updated on 9/17/2020 at 10:53:07 PM

Append new data

Overwrite table with new data

All existing data will be deleted from the table whether or not the loading action completes successfully.

| COLUMN    | DATA TYPE | NULLABLE | NAME |
|-----------|-----------|----------|------|
| ORGANI... | INTEGER   |          |      |
| ORDER_... | INTEGER   |          |      |
| RTL...    | INTEGER   |          |      |

Back Next

**Note**

You use the value for <db2-username> as the schema because Db2 uses the username as the default schema for all unqualified tables.

- 26. Review the data and switch off the **Header in first row** option.

| ORGANIZATION_KEY<br>INTEGER | ORDER_DAY_KEY<br>INTEGER | RTL_COUNTRY_KEY<br>INTEGER | EMPLOYEE_KEY<br>INTEGER | RETAILER_KEY<br>INTEGER | PRODUCT_K<br>INTEGER |
|-----------------------------|--------------------------|----------------------------|-------------------------|-------------------------|----------------------|
| 1 11121                     | 20040112                 | 90010                      | 4008                    | 7166                    | 30002                |
| 2 11121                     | 20040112                 | 90010                      | 4008                    | 7166                    | 30003                |

- 27. Click **Next**.

- 28. Click **Begin Load**.

**LOAD DATA**

You are loading the file **GOSALES DW.MRK\_PROMOTION\_FACT.txt** into **SMF35487.MRK\_PROMOTION\_FACT**

**Review settings**

|                                                        |                            |
|--------------------------------------------------------|----------------------------|
| <b>Summary</b>                                         | <b>Option</b>              |
| <b>Code page:</b> 1208 (Default)                       | Maximum number of warnings |
| <b>Separator:</b> , (Default)                          | 1000                       |
| <b>Header in first row:</b> No                         |                            |
| <b>Time format:</b> HH:MM:SS (Default)                 |                            |
| <b>Date format:</b> YYYY-MM-DD (Default)               |                            |
| <b>Timestamp format:</b> YYYY-MM-DD HH:MM:SS (Default) |                            |
| <b>String delimiter:</b> (Default)                     |                            |
| <b>Back</b>                                            | <b>Begin Load</b>          |

- 29. When the Load completes, review the Load details. Verify that the number of **Rows loaded** is 11,034.

Load details

| Rows read | Rows loaded | Rows rejected |
|-----------|-------------|---------------|
| 11,034    | 11,034      | 0             |

Start time  
09/17/2020 11:53:03 PM

End time  
09/17/2020 11:53:05 PM

The data load job succeeded.  
You can now work with your data.

No errors



**Stop**

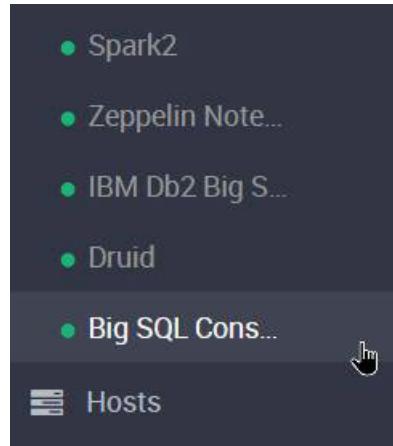
You must have special privileges that are granted to your <username> before you can proceed with Part 2. Let the instructor know that you are ready to start Part 2 before you proceed.

## Part 2: Creating a server object

The server object maps to exactly one database. The CREATE SERVER statement defines a data source. The CREATE SERVER statement requires the definition of a server type.

In this part, you create a server object for the Db2 on IBM Cloud database, which defines it to the federated Db2 Big SQL server running on your VM.

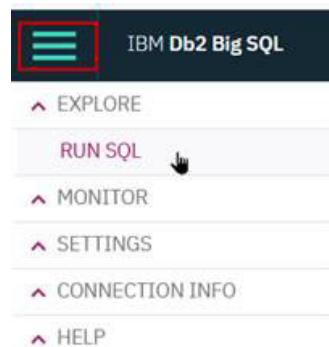
- 1. Open the Ambari Web UI by entering the Ambari URL value <hostname:8080> in your browser. Log in with the Ambari Username <ambari username> and the Ambari Password <ambari password>.
- 2. Select **Big SQL Console** from the menu on the left.



- \_\_\_ 3. Select **Console UI** from the Quick Links menu.

A screenshot of the IBM Db2 Big SQL interface. At the top, there are tabs for 'SUMMARY' (which is selected) and 'CONFIGS'. On the right, there is a 'ACTIONS' button with a dropdown arrow. The main area shows a 'Summary' section with a 'Components' table. The table has one row with 'Started' status and a link to 'BIG SQL CONSOLE'. To the right of the summary is a 'Quick Links' sidebar with a single item: 'Console UI'.

- \_\_\_ 4. Log in with your <username> and <password>.  
\_\_\_ 5. Click **Run SQL** to open the canvas.



- \_\_\_ 6. Create the server. Enter the following statement.

```
create server <db2_on_cloud> type DB2/UDB version 11
authorization <db2-username> password "<db2-password>"
options (host '<db2-hostname>', port '<db2-port>', dbname '<db2-db>',
password 'Y');
```

**Note**

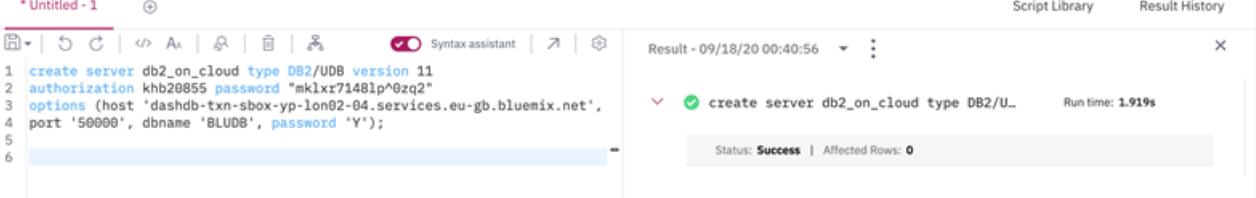
<db2\_on\_cloud> is the server name, which must be unique. For example, you can name the server db2\_on\_cloud\_<username>. For example, db2\_on\_cloud\_student0010.

In this exercise, replace <db2\_on\_cloud> by your server name, that is, the value that you choose in this step.

---

7. Run the statement.

RUN SQL



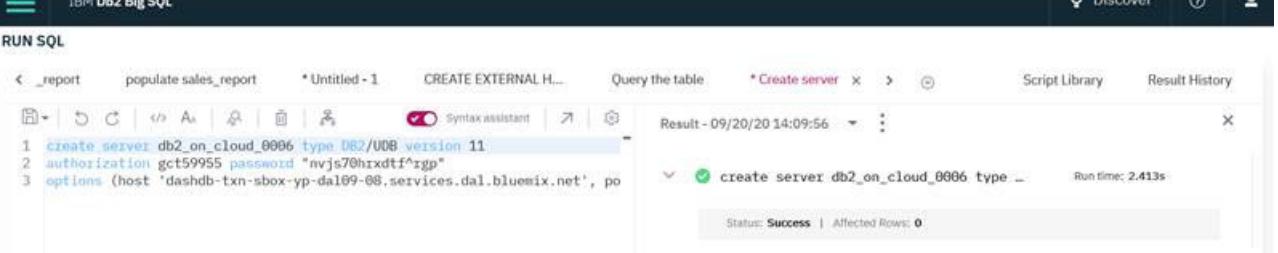
```
* Untitled - 1
1 create server db2_on_cloud type DB2/UDB version 11
2 authorization khb20855 password "mklxr7148lp^0zq2"
3 options (host 'dashdb-txn-sbox-yp-lon02-04.services.eu-gb.bluemix.net',
4 port '50000', dbname 'BLUDB', password 'Y');
5
6
```

Result - 09/18/20 00:40:56

create server db2\_on\_cloud type DB2/U... Run time: 1.919s

Status: Success | Affected Rows: 0

The expected result is similar to the following output.



IBM Db2 Big SQL

RUN SQL

```
_report populate sales_report * Untitled - 1 CREATE EXTERNAL H... Query the table * Create server x > ⚡ Script Library Result History
```

```
1 create server db2_on_cloud_0006 type DB2/UDB version 11
2 authorization gct59955 password "nvjs70hixdti^rgp"
3 options (host 'dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net', po
```

Result - 09/20/20 14:09:56

create server db2\_on\_cloud\_0006 type ... Run time: 2.413s

Status: Success | Affected Rows: 0

### Part 3: Creating the user mapping object

A user mapping is an association between an authorization ID on the federated server and the information that is required to connect to the remote data source.

To create a user mapping, you specify the following information in the **CREATE USER MAPPING** statement:

- local authorization ID (your <username>)
- local name of the remote data source server as specified in the server definition (your <db2\_on\_cloud>)
- remote ID and password (<db2-username>) and <db2-password>)

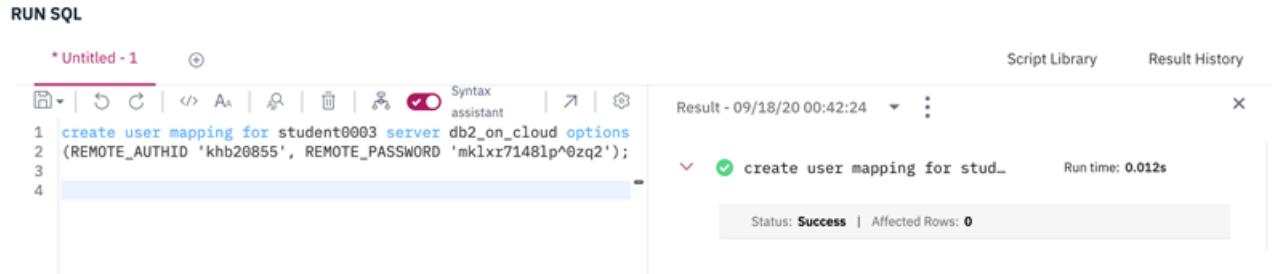
By default, the federated server stores user mapping in the SYSCAT.USEROPTIONS view in the global catalog and encrypts the remote passwords.

In this part, you map your <username> in the federated Db2 Big SQL server running on the VM to your <db2-username> for your Db2 service instance on IBM Cloud.

- 1. On the same canvas, create the mapping by entering the following statement:

```
create user mapping for <username> server <db2_on_cloud> options
(REMOTE_AUTHID '<db2-username>', REMOTE_PASSWORD '<db2-password>');
```

- 2. Run the statement.



The screenshot shows the RUN SQL interface with a script editor containing the following SQL statement:

```
1 create user mapping for student0003 server db2_on_cloud options
2 (REMOTE_AUTHID 'khb20855', REMOTE_PASSWORD 'mk1xr7148lp^0zq2');
3
```

The status bar at the bottom indicates "Status: Success | Affected Rows: 0".

## Part 4: Creating the nickname

A nickname is an identifier that an application uses to reference a data source object, such as a table or view. A nickname represents a remote table or view.

The simplest syntax for the CREATE NICKNAME statement is as follows:

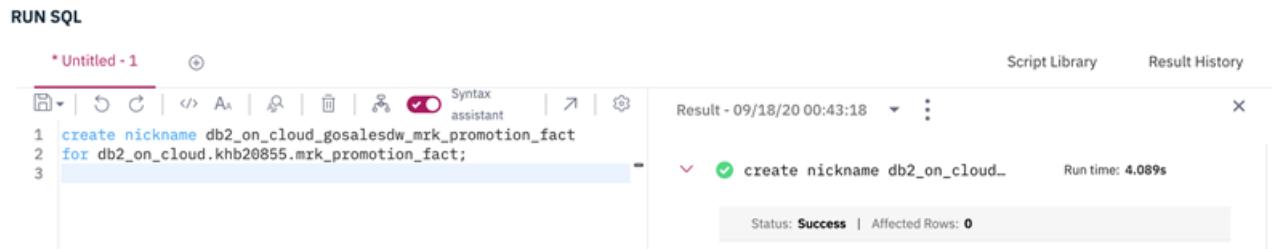
```
CREATE NICKNAME NICKNAME <nickname> FOR <remote-object-name> OPTIONS ....
```

In this part, you create a nickname for the table `mrk_promotion_fact` that you added to Db2 on Cloud in Part 1.

- 1. Enter the CREATE NICKNAME statement.

```
create nickname db2_on_cloud_gosalesdw_mrk_promotion_fact for
<db2_on_cloud>.<db2-username>.mrk_promotion_fact;
```

- 2. Run the statement.



The screenshot shows the RUN SQL interface with a script editor containing the following SQL statement:

```
1 create nickname db2_on_cloud_gosalesdw_mrk_promotion_fact
2 for db2_on_cloud.khb20855.mrk_promotion_fact;
3
```

The status bar at the bottom indicates "Status: Success | Affected Rows: 0".

## Part 5: Running queries

When you want to select or modify data source data, you query the nicknames by using the SELECT, INSERT, UPDATE, and DELETE statements. You submit queries in SQL to the federated database.

In this part, you use the SELECT statement to query the nickname of the table in the remote data source on Db2 on Cloud. You submit the query to the federated server on your VM by using the Db2 Big SQL console.

- 1. Select from the table from within Db2 Big SQL. Enter in the query.

```
select * from db2_on_cloud_gosalesdw_mrk_promotion_fact;
```

Result - 09/18/20 00:23:58

The screenshot shows the Db2 Big SQL console interface. At the top, it displays the query: "select \* from db2\_on\_cloud\_gosalesdw\_mrk\_promotion\_fact" and the run time: "Run time: 5.043s". Below this, the result set is shown in a table format with the following columns: ORGANIZATION\_KEY, ORDER\_DAY\_KEY, RTL\_COUNTRY\_KEY, EMPLOYEE\_KEY, RETAILER\_KEY, and PRODUCT\_KEY. The data consists of 12 rows of sample data.

| ORGANIZATION_KEY | ORDER_DAY_KEY | RTL_COUNTRY_KEY | EMPLOYEE_KEY | RETAILER_KEY | PRODUCT_KEY |
|------------------|---------------|-----------------|--------------|--------------|-------------|
| 11121            | 20040112      | 90010           | 4008         | 7166         | 30002       |
| 11121            | 20040112      | 90010           | 4008         | 7166         | 30003       |
| 11121            | 20040112      | 90011           | 4011         | 6838         | 30002       |
| 11121            | 20040112      | 90012           | 4013         | 6737         | 30003       |
| 11121            | 20040112      | 90013           | 4045         | 7266         | 30002       |
| 11121            | 20040112      | 90013           | 4045         | 7266         | 30003       |
| 11121            | 20040209      | 90010           | 4008         | 6846         | 30008       |
| 11121            | 20040209      | 90010           | 4069         | 7162         | 30003       |
| 11121            | 20040209      | 90010           | 4069         | 7164         | 30007       |
| 11121            | 20040209      | 90011           | 4011         | 6838         | 30008       |
| 11121            | 20040209      | 90011           | 4078         | 7238         | 30007       |

- 2. You can try the same query with the three part name and you get the same results. The syntax for a three part name is: <server\_name>.<schema>.<table>.

```
select * from <db2_on_cloud>.<db2-username>.mrk_promotion_fact;
```

Result - 09/18/20 00:50:29

The screenshot shows the Db2 Big SQL console interface. At the top, it displays the query: "select \* from db2\_on\_cloud.khb20855.mrk\_promotion\_fact" and the run time: "Run time: 4.023s". Below this, the result set is shown in a table format with the following columns: ORGANIZATION\_KEY, ORDER\_DAY\_KEY, RTL\_COUNTRY\_KEY, EMPLOYEE\_KEY, RETAILER\_KEY, and PRODUCT\_KEY. The data consists of 12 rows of sample data.

| ORGANIZATION_KEY | ORDER_DAY_KEY | RTL_COUNTRY_KEY | EMPLOYEE_KEY | RETAILER_KEY | PRODUCT_KEY |
|------------------|---------------|-----------------|--------------|--------------|-------------|
| 11121            | 20040112      | 90010           | 4008         | 7166         | 30002       |
| 11121            | 20040112      | 90010           | 4008         | 7166         | 30003       |
| 11121            | 20040112      | 90011           | 4011         | 6838         | 30002       |
| 11121            | 20040112      | 90012           | 4013         | 6737         | 30003       |
| 11121            | 20040112      | 90013           | 4045         | 7266         | 30002       |

The three-part name is useful if you have many federated objects and you do not want to manually manage the nicknames when a schema changes.

## End of exercise

## Exercise review and wrap-up

In this exercise, you followed the basic steps to get started with the federation feature of Db2 Big SQL. You created a Db2 service instance on IBM Cloud to represent a remote data source. You created a server object, a user mapping object, and a nickname. Finally, you queried the table on the remote data source from the federated Db2 Big SQL server on your VM by using both the nickname and the three-part name (SERVER.REMOTE\_USER.REMOTE\_TABLE).



IBM Training



© Copyright International Business Machines Corporation 2016, 2021.