Cloudera Runtime 7.0.3

Using Phoenix to Store and Access Data

Date published: 2019-11-22
Date modified:

https://docs.cloudera.com/

Legal Notice

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Mapping Phoenix Schemas to HBase Namespaces

You can map a Phoenix schema to an HBase namespace to gain multitenancy features in Phoenix.

⚠ **Important:** You must use this feature only in a CDP Private Cloud Base deployment. This feature is configured automatically in a CDP Public Cloud deployment.

HBase, the underlying storage engine for Phoenix, has namespaces to support multi-tenancy features. Multitenancy helps an HBase user or administrator to perform access control and quota management tasks. Also, namespaces enable tighter control of where a particular data set is stored on RegionsServers.

Enable namespace mapping

You can enable namespace mapping by configuring a set of properties using Cloudera Manager.

About this task

After you set the properties to enable the mapping of Phoenix schemas to HBase namespaces, reverting the property settings renders the Phoenix database unusable. Test or carefully plan the Phoenix to HBase namespace mappings before implementing them.

⚠ **Important:** Cloudera recommends that you enable namespace mapping. If you decide not to enable this feature, you can skip the following steps.

To enable Phoenix schema mapping to a non-default HBase namespace:

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope (Service-Wide) .
4. Locate the HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.
5. Add the following property values:

   Name: phoenix.schema.isNamespaceMappingEnabled

   Description: Enables mapping of tables of a Phoenix schema to a non-default HBase namespace. To enable mapping of a schema to a non-default namespace, set the value of this property to true. The default setting for this property is false.

   Value: true

   Name: phoenix.schema.mapSystemTablesToNamespace

   Description: With true setting (default): After namespace mapping is enabled with the other property, all system tables, if any, are migrated to a namespace called system. With false setting: System tables are associated with the default namespace.

   Value: true
6. Select Scope Gateway .
7. Locate the HBase Client Advanced Configuration Snippet (Safety Valve) for hbase-site.xml property or search for it by typing its name in the Search box.

8. Add the following property values:

   Name: phoenix.schema.isNamespaceMappingEnabled

   Description: Enables mapping of tables of a Phoenix schema to a non-default HBase namespace. To enable mapping of the schema to a non-default namespace, set the value of this property to true. The default setting for this property is false.

   Value: true

   Name: phoenix.schema.mapSystemTablesToNamespace

   Description: With true setting (default): After namespace mapping is enabled with the other property, all system tables, if any, are migrated to a namespace called system.With false setting: System tables are associated with the default namespace.

   Value: true
9. Enter a Reason for change, and then click Save Changes to commit the changes.
10. Restart the role and service when Cloudera Manager prompts you to restart.

    Note: If you do not want to map Phoenix system tables to namespaces because of compatibility issues with your current applications, set the phoenix.schema.mapSystemTablesToNamespace property to false.

Associating Tables of a Schema to a Namespace

Important: You must use this feature only in a CDP Private Cloud Base deployment. This feature is configured automatically in a CDP Public Cloud deployment.

After you enable namespace mapping on a Phoenix schema that already has tables, you can migrate the tables to an HBase namespace. The namespace directory that contains the migrated tables inherits the schema name.

For example, if the schema name is store1, then the full path to the namespace is $hbase.rootdir/data/store1. System tables are migrated to the namespace automatically during the first connection after enabling namespace properties.

Associate table in a customized Kerberos environment

You can run a command to associate a table in a customized environment without Kerberos.

Before you begin

In a Kerberos-secured environment, you must have admin privileges (user hbase) to complete the following task.

Procedure

- Run a command to migrate a table of a schema to a namespace, using the following command syntax for the options that apply to your environment:

```
phoenix-psql
ZooKeeper_hostnames:2181
:zookeeper.znode.parent
:HBase_headless_keytab_location
:principal_name
;TenantId=tenant_Id
;CurrentSCN=current_SCN
-m
schema_name.table_name
```

Associate a table in a non-customized environment without Kerberos

You can run a command to associate a table in a non-customized environment without Kerberos.

Procedure

- Run the following command to associate a table:

```
phoenix-psql ZooKeeper_hostname -m Schema_name.table_name
```

Orchestrating SQL and APIs with Apache Phoenix

You can use the standard JDBC APIs instead of the regular HBase client APIs to create tables, insert data, and query your HBase data.

The following drivers are currently supported:

- JDBC driver
- Microsoft .Net driver
- Other non-Java drivers (example, Python driver)

Obtaining a driver for application development

Based on your application development requirements, you can obtain one of the following drivers:

JDBC driver

Use the /opt/cloudera/parcels/<*phoenix parcel*>/lib/phoenix/<*phoenix parcel*>-client.jar file in the Phoenix server-client repository. If you use the repository, download the JAR file corresponding to your installed CDH version.

JDBC driver as a Maven dependency

You can pull the CDH 6 artifacts from the Cloudera Maven repository from here: https://archive.cloudera.com/p/phoenix/6.2/maven-repository/. You can access the Maven repository using your Enterprise Support Subscription credentials.

Microsoft .NET Driver

Download and install a NuGet package for Microsoft .NET Driver for Apache Phoenix. for more information, see https://www.nuget.org/packages/Microsoft.Phoenix.Client/.

Other non-Java drivers

Other non-JDBC Drivers for Phoenix are available as CDH add-ons and on other websites, but they are not currently supported by Cloudera. You can find compatible client drivers by constructing a web search string consisting of "avatica" and the name of an application programming language that you want to use. Example: avatica python.

For more information about obtaining drivers for your application development, see http://phoenix.apache.org/faq.html.

Configure Phoenix Query Server

Before you begin

The HBase configuration provides most of the settings that enable secure Kerberos environments for Phoenix. However, there are additional configuration properties that complete the setup of Kerberos security for the Phoenix Query Server.

To configure Phoenix Query Server using Cloudera Manager:

Procedure

1. Go to the HBase service.
2. Click the Configuration tab.
3. Select Scope *HBase Cluster* (Service-Wide) .
4. Select Category Security .
5. Locate the HBase Secure Authentication property or search for it by typing its name in the Search box.
6. Select Kerberos.
7. Enter a Reason for change, and then click Save Changes to commit the changes.
8. Restart the role and service when Cloudera Manager prompts you to restart.

Thin Client connectivity with Phoenix Query Server

The Phoenix Query Server (PQS) is a component of the Apache Phoenix distribution. PQS provides an alternative means to connect directly. The thin clients access the PQS using HTTP, and the PQS executes the statements for the thin clients, and returns the results.

This topology offloads most computation to PQS and requires a smaller client-side footprint. The PQS client protocol is based on the Avatica component of Apache Calcite.

Understanding User-Defined Functions (UDFs) in Phoenix

With a user-defined function (UDF), you can extend the functionality of your SQL statements by creating scalar functions that operate on a specific tenant. You can create temporary or permanent user-defined or domain-specific scalar functions. The UDFs can be used as built-in functions in queries such as SELECT, UPSERT, and DELETE. Temporary functions are specific to a session/connection and are not accessible in other sessions/connections. Meta-information of permanent functions will be stored in the system table called SYSTEM.FUNCTION. Tenant specific functions are supported, and functions created in a tenant-specific connection is not visible to other tenant-specific connections. Only global tenant-specific functions are visible to all the connections.

For details about creating, dropping, and how to use UDFs for Phoenix, see User-defined functions on the Apache website here: https://phoenix.apache.org/udf.html.

Use the Phoenix client to load data

You must use the Phoenix client to load data into the HBase database and also to write to the Phoenix tables.

Index updates are automatically generated by the Phoenix client and there is no user intervention or effort required. Whenever a record is written to the Phoenix tables, the client generates the updates for the indexes automatically.

Note: If the Phoenix table has indexes, you can use the JDBC driver or CSV bulk load table to update or ingest data.

It is highly recommended that you use the Phoenix client to load data into the HBase database and also to write to the Phoenix tables. If you use the HBase APIs to write data to a Phoenix data table, the indexes for that Phoenix data table will not be updated.

Use indexes in Phoenix

Apache Phoenix automatically uses indexes to service a query. Phoenix supports global and local indexes. Each is useful in specific scenarios and has its own performance characteristics.

Global indexes in Phoenix

You can use global indexes for READ-heavy use cases. Each global index is stored in its own table and thus is not co-located with the data table. With global indexes, you can disperse the READ load between the main and secondary index table on different RegionServers serving different sets of access patterns. A Global index is a covered index. It is used for queries only when all columns in that query are included in that index.

Local indexes in Phoenix

You can use local indexes for WRITE-heavy use cases. Each local index is stored within the data table. With global indexes, you can use local indexes even when all columns referenced in a query are not contained in the index.

This is done by default for local indexes because the table and index data reside on the same region server and hence it ensures that the lookup is local.