

## Operational Database Overview

Date published: 2020-02-29

Date modified: 2023-06-05

# CLOUDERA

# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

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

<b>Operational Database overview.....</b>	<b>4</b>
Introduction to Apache HBase.....	6
Introduction to Apache Phoenix.....	6
Apache Phoenix and SQL.....	6

## Operational Database overview

Cloudera Operational Database delivers a real-time, always available, scalable operational database that serves traditional structured data alongside unstructured data within a unified operational and warehousing platform.

Cloudera Operational Database is powered by Apache HBase and Apache Phoenix. In Cloudera Operational Database, you use Apache HBase as a datastore with HDFS and/or S3 providing the storage infrastructure. You have the choice to either develop applications using one of the native Apache HBase API, or you can use Apache Phoenix for data access. Apache Phoenix is a SQL layer that provides a programmatic ANSI SQL interface. It works on top of Apache HBase, and it makes it possible to handle data using standard SQL queries and Apache Phoenix commands. You can use Cloudera Operational Database in the public cloud or on-premises.

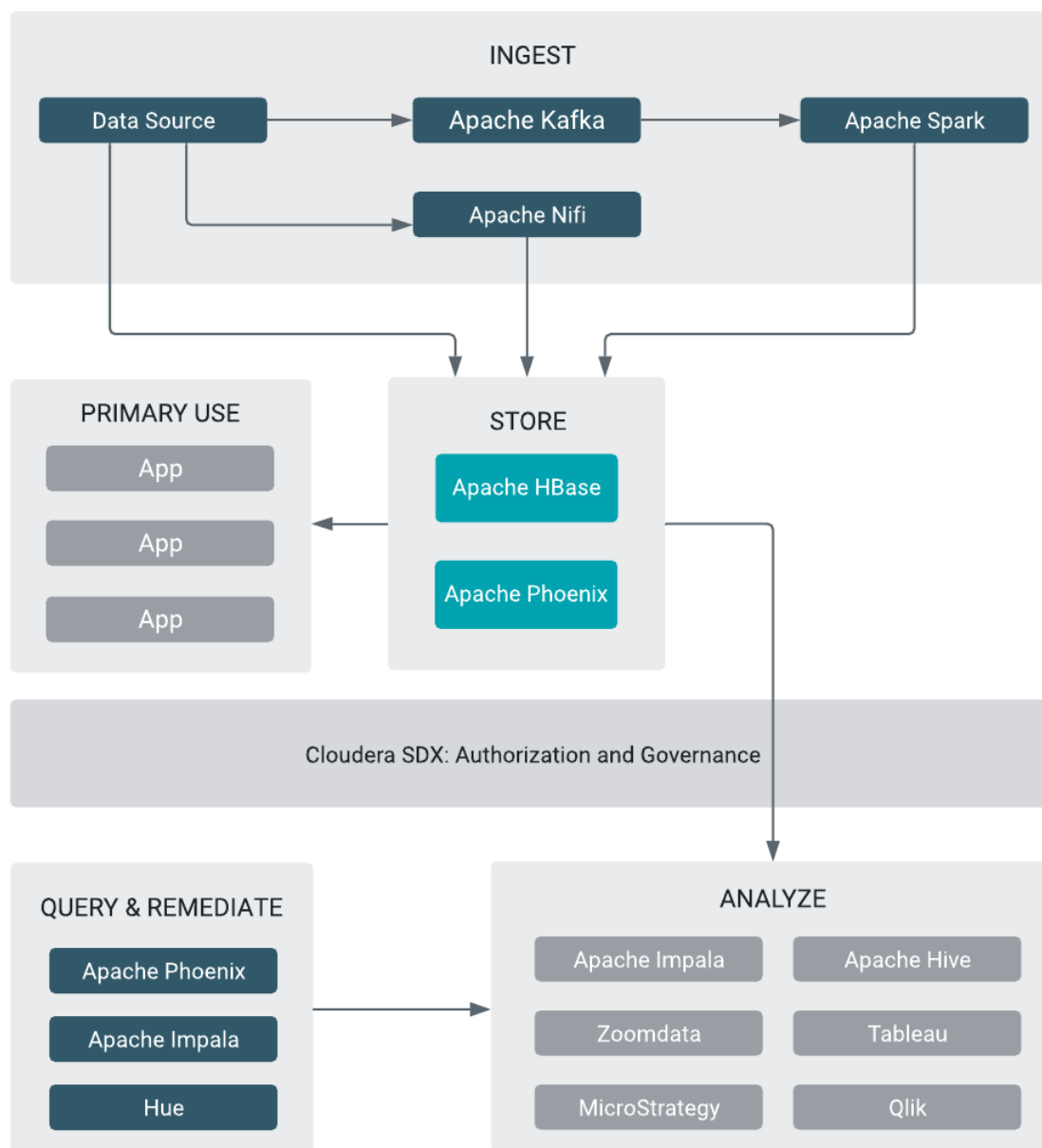
### Public cloud

- CDP Data Hub Operational Database template
- Cloudera Operational Database (COD) experience that is a managed database platform as a service (dbPaaS)

### On-premises

- CDP Private Cloud Base

Cloudera Operational Database plays the crucial role of a data store in the enterprise data lifecycle. The following graphic shows some of the key elements in a typical operational database deployment.



The operational database has the following components:

- Apache Phoenix provides a SQL interface that runs on top of Apache HBase.
- Apache HBase provides the key-value stores with massive scalability, so you can store unlimited amounts of data in a single platform and handle growing demands for serving data.
- Apache ZooKeeper provides a distributed configuration service, a synchronization service, and a naming registry.
- Apache Knox Gateway provides perimeter security so that the enterprise can confidently extend access to new users.
- Apache HDFS is used to write the Apache HBase WALs.
- Hue provides a web-based editor to create and browse Apache HBase tables.
- Object store such as Amazon S3 and Microsoft ADLS Gen2 is used to store the Apache HBase HFiles.

- CCloudera Shared Data Experience (SDX) is used for security and governance capabilities. Security and governance policies are set once and applied across all data and workloads.
- IDBroker provides an authentication mechanism that is built as part of the Apache Knox authentication service. It allows an authenticated and authorized user to exchange a set of credentials or a token for cloud vendor access tokens.

## Introduction to Apache HBase

Apache HBase is a scalable, distributed, column-oriented datastore. Apache HBase provides real-time read/write random access to very large datasets hosted on HDFS.

HBase is sparse, distributed, persistent, multidimensional sorted map, which is indexed by rowkey, column key, and timestamp. Fundamentally, it is a platform for storing and retrieving data with random access. HBase is designed to run on a cluster of computers, built using commodity hardware. HBase is also horizontally scalable, which means you can add any number of columns anytime.

## Introduction to Apache Phoenix

Apache Phoenix is a SQL layer for Apache HBase that provides a programmatic ANSI SQL interface.

Apache Phoenix implements best-practice optimizations to enable software engineers to develop HBase based next-generation applications that operationalize big data. Using Phoenix, you can create and interact with tables in the form of typical DDL/DML statements using the Phoenix standard JDBC API.

Apache HBase timestamps are the core part of the Apache Phoenix functionality. It is used as part of the *Travel in time* functionality and by the Transactional engines. Due to this, we expect that the data must have valid timestamps that signifies the time when the record is created. Invalid timestamps cause incorrect behavior for most Apache Phoenix functionalities, such as secondary indexes or CRUD operations. If your data is relying on the special behavior of the record's timestamps or future timestamps, you must not use such data with Apache Phoenix.

## Apache Phoenix and SQL

You can use the Apache Phoenix SQL commands to create, drop, or modify an Apache HBase table. You can also create Apache Phoenix views that are virtual tables that share the same Apache HBase table.

The Apache Phoenix commands enable you to use standard SQL data definition language (DDL) and data manipulation language (DML) commands to interact with Apache HBase tables. You can create a new Apache HBase table using the standard `CREATE TABLE` SQL command or create a view on an existing Apache HBase table using the `VIEW` command. View enables you to have multiple virtual tables that share the same physical Apache HBase table.

Apache HBase tables and Apache Phoenix tables have a one-to-one relationship. This means each Apache HBase table is associated with a corresponding Apache Phoenix table.



**Warning:** Although you can modify Apache Phoenix tables using the Apache HBase native APIs, this is not supported and results in errors, inconsistent indexes, incorrect query results, and sometimes corrupt data.

Using Apache Phoenix commands you can do the following tasks:

- Create or alter Apache HBase tables using DDL commands like `CREATE TABLE`
- Modify contents in an Apache HBase table using DML commands like `UPSERT VALUES`