

Using Materialized Views

Date published: 2019-12-17

Date modified: 2022-05-05



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

Introduction to Materialized Views.....	4
Creating Materialized Views.....	4
Configuring Retention Time for Materialized Views.....	10
Using Dynamic Materialized View Endpoints.....	13
Using SQL Stream Builder with Cloudera Data Visualization.....	16

Introduction to Materialized Views

SQL Stream Builder has the capability to materialize results from a Streaming SQL query to a persistent view of the data that can be read through REST. Business Intelligence tools and applications can use the Materialized View REST endpoint to query streams of data without deploying database systems.

Materialized Views are in synchronization with the mutating stream - they are updated by a primary key as data flows through the system. The data is updated by a given key, and it represents the latest view of the data by key.

For example: vehicleID Z latest latitude and longitude is X and Y. As the vehicle moves, the latitude and longitude for the vehicleID are updated. The primary key is defined at creation time and is immutable.

Materialized Views can be created as mutating snapshots of the queried data result that is updated by a given key. The data is always the latest representation of itself by key (analogous to a primary key in most RDBMS systems).

You can query the Materialized Views using a GET request over REST, which returns a JSON response as "Content-Type: application/json". The queries are not defined at query time. Rather, they are curated, saved, and granted access through the Cloudera platform. You can configure a REST endpoint to query the Materialized View. Multiple query conditions can be created to allow various ways to query the same data. This is sometimes referred to as a 'pull query'.

Null Keys

When a key is removed from the incoming messages of a source, SSB continues to consume them. However, it marks the missing key as NULL at the sink. Similarly, when a key is removed from the source schema, but not from the incoming messages of the source, SSB ignores the key on the incoming stream.

Creating Materialized Views

After executing a SQL Stream job, you can set up the Materialized Views to have a snapshot of your queried data. You can use the URL Pattern from the Materialized View to visualize the generated data.

Before you begin

- Make sure that PostgreSQL is installed and configured to SQL Stream Builder (SSB) to create Materialized Views.

Procedure

1. Navigate to the Streaming SQL Console.
 - a) Go to your cluster in Cloudera Manager.
 - b) Select SQL Stream Builder from the list of services.
 - c) Click SQLStreamBuilder Console.

The **Streaming SQL Console** opens in a new window.

2. Click Create Job or select a previous job on the **Getting Started** page.

You are redirected to the **Console** page. The Materialized View button will be available when you add a query to the SQL Editor.

3. Click Materialized View.

Materialized View

Configuration

Please select or create an API Key

Primary Key

Retention (Seconds)

300

Min Row Retention Count

API Key

+

Enable MV

Recreate on Job Start

Ignore NULLs

Queries

+ Add New Query

No Queries

4. Switch Enable MV toggle.

5. Select a Primary Key.

If this list is empty, then no SQL is specified in the **SQL Editor** or the SQL query is invalid. Select a key as a primary key for the Materialized View. All data will be updated by this key.

6. Select a Retention Period.

Data not being mutated during this period is removed from the view.

7. Enable or disable Recreate on Job Start.

If enabled, the Materialized View is deleted when a job is started or restarted.

8. Enable or disable Ignore NULLS.

If enabled, NULL values will NOT update values that are non null - they are ignored.

9. Select an API Key.

In case there are no API Keys, click Add API key, or click Materialized Views from the main menu. The add API key window appears. Provide a name for the API key, and click Save Changes.

API Key

×

Key Value

0862eb36-70cf-4ad9-840a-0a3d175cc4ca

Key Name

Cancel

Save

To check your created API keys:

a. Click Materialized Views from the main menu.

The Materialized Views page appears.

b. Select API Keys tab.**10. Click Add New Query to create the Materialized View query.**

The Materialized View Query Configuration window appears.

Query Editor

×

URL /api/v1/query/5195/

URL Pattern * Description (Optional)

Columns Filters

Select Column

Select Column

Select All

Unselect All

⚠ No Columns Selected

Cancel

Apply

Apply and Save Job

11. Provide a name in the **URL Pattern** field.
12. Provide a description of the Materialized View Query, if needed.

13. Customize the Materialized View in the Query Editor.

Query Editor

URL

/api/v1/query/5195/test

URL Pattern *

test

Description (Optional)

Columns

Filters

Select Column

Select Column

Select All

Unselect All

Name	Alias	Type	
col_str	col_str	VARCHAR	
col_int	col_int	INTEGER	
col_ts	col_ts	TIMESTAMP_WITHOUT_TIME_ZONE	

Cancel

Apply

Apply and Save Job

- a) Select the columns of the SQL job you want to use in the Materialized View Query.
You can select every column in the table by clicking on the Select All button.
- b) Click Filters tab to apply computations and further enrichment of your data.

Query Editor ×

URL /api/v1/query/5195/test

URL Pattern * Description (Optional)

test

Columns Filters

AND
OR

+ Rule
+ Ruleset

Field	Operator	Value
col_str	equal	

Cancel
Apply
Apply and Save Job

c) Provide details of the filter by selecting the Field, Operator and Value.

You can add more rules or set a ruleset for the query.

14. Click Apply or Apply and Save Job.

The URL for the Materialized View appears under the **Queries** header.

Materialized View

Configuration

Primary Key

col_str

Retention (Seconds)

300

Min Row Retention Count

API Key

docstest

+
✎
✖

☒ Enable MV

☐ Recreate on Job Start

☐ Ignore NULLs

Queries ✎

/api/v1/query/5195/test?key=0862eb36-70cf-4ad9-840a-0a3d175cc4ca

+
✎
✖

Results

You can click the created REST endpoint to review the data, or copy it and visualize the queried data in a Business Intelligence tool, notebook, code and so on. You can also review the list of Materialized View and API keys on the Materialized View page.

Related Information

[Configuring databases for SSB](#)

[Running a SQL Stream job](#)

Configuring Retention Time for Materialized Views

When creating Materialized Views, you can configure how the system should retain the data rows in the Materialized Views. You can either choose between retaining the data by time or the row count.

The Materialized Views configuration allows you to set one of the following configuration parameters for data retention:

- Retention Time
- Min Row Retention Count

You can specify the Retention Time and Min Row Retention Count when creating a Materialized View for a SQL job on the Compose page of the Streaming SQL Console.

Retention Time

Retention Time is specified in seconds, and it tells the system to retain data rows as old as the specified retention time. The rows that are outside of the retention time are removed from the Materialized View.



Note: You can only add a Retention Time value, if the Min Row Retention Count field is empty or set to 0.

The following example shows how to set a Retention Time of 300 seconds. This means that only those rows are included in the Materialized View that are within the 300 seconds of the job execution. The older rows are removed from the Materialized Views.

>>

Materialized View

Configuration

Primary Key

col_int

Retention (Seconds)

300

Min Row Retention Count

API Key

SecureKey

+

Enable MV

Recreate on Job Start

Ignore NULLs

Queries

/query/5195/test?key=6c535ed3-9668-464a-a4ad-542c61d8e0d5

Minimum Row Retention Count

The Min Row Retention Count parameter indicated to the system to maintain a specific number of data rows in the Materialized View.



Note: You can only add a Min Row Retention Count value, if the Retention Time field is empty or set to 0.

The following example shows how to configure the system to retain the last 5000 data rows. This means that only first 5000 data rows are included in the Materialized View, and the data rows from 5001 are removed from the Materialized View.

Materialized View

Configuration

Primary Key

col_int

Retention (Seconds)

0

Min Row Retention Count

5000

API Key

SecureKey

Enable MV

Recreate on Job Start

Ignore NULLs

+

Queries

/query/5195/test?key=6c535ed3-9668-464a-a4ad-542c61d8e0d5

Retaining all data without limit

In order to retain all data rows, regardless of time, or number of rows in the Materialized View, both settings can be reset to zero (0). This indicates to the system that all data must be preserved without a time limit.

The following example shows how to configure the Retention parameters to keep all of the data regardless of time:

>>

Materialized View

Configuration

Primary Key

col_int

Retention (Seconds)

0

Min Row Retention Count

0

API Key

SecureKey

+

Enable MV

Recreate on Job Start

Ignore NULLs

Queries

/query/5195/test?key=6c535ed3-9668-464a-a4ad-542c61d8e0d5

Using Dynamic Materialized View Endpoints

You can use static or dynamic REST endpoints when creating Materialized Views in SQL Stream Builder. After setting filters for the Materialized View query, you can further filter down the results by using variables in the endpoint URL.

Difference between static and dynamic endpoints

When using a static endpoint, the endpoint URL serves as a constant string, the process of filtering is constant and final. The results are displayed based on the configurations and filters of the query. In case of a static endpoint, you provide the information in the URL pattern field only with a reference purpose. For example, in the following static endpoint the Filter is set to *id greater than 0*, and the endpoint URL is *positive/id*:

13

Query Editor

**URL** `/api/v1/query/5197/positive/id`

URL Pattern * ⓘ

positive/id

Description (Optional)

Columns **Filters**

AND OR

+ Rule

+ Ruleset

Field

id

Operator

greater

Value

0

Cancel

Apply

Apply and Save Job

In case of a dynamic endpoint, you have the option to set dynamic variables in the endpoint URL. This means that when providing the information in the URL pattern field, you also include a generic variable that later can be specified when using the REST endpoint. With dynamic endpoints, you can further filter the queried results based on the given variable. When creating the dynamic endpoint, you need to make sure that the variable is surrounded by curly braces to signal that it is a variable as *minimumId* in the example shown below. You also need to reference the variable in the same way when setting the Filters for the Materialized View Query. For example, in the following dynamic endpoint the Filter is set to *id greater than {minimumId}*, and the endpoint URL is *id/bigger/than/{minimumId}*:

Query Editor



URL

`/api/v1/query/5197/id/bigger/than/{minimumId}`

URL Pattern * ⓘ

`id/bigger/than/{minimumId}`

Description (Optional)

Columns **Filters**

AND

OR

+ Rule

+ Ruleset

Field

id

Operator

greater

Value

{minimumId}

Cancel

Apply

Apply and Save Job

Using dynamic endpoints

You can use a static endpoint by clicking on it on the Materialized View page or by copying the URL and pasting it to a browser, application or tool. As the endpoint is static, the URL can be invoked and the aspects of filtering is constant. However, as a dynamic endpoint contains a variable, you need to replace it by an actual value. This value makes the endpoint and the filtering dynamic. In the following example, a static and dynamic endpoint is compared:

URL Pattern

Description

`/api/v1/query/5424/id/bigger/than/{minimumId}?key=225342de-8588-4db9-8bfc-e52d9dccf34f``/api/v1/query/5424/postive/id?key=225342de-8588-4db9-8bfc-e52d9dccf34f`

When you click on a dynamic endpoint, you are prompted to provide a value for the variable as shown in the following illustration:

Dynamic endpoint

Please specify the values of the variables in this dynamic endpoint:

minimumId

Close

Go

After entering a value, the filtered results are shown in a new browser tab:

```

<  →  ↻  Not Secure | mgergely-1.vpc.cloudera.com:18131/api/v1/query/5424/id/bigger/than/0?key=225342de-8588-4db9-8bfc-e52d9dccc34f
[{"id":"8","power":null,"age":"561"}, {"id":"5","power":"Endurance","age":"390"}, {"id":"3","power":null,"age":"29"}, {"id":"4","power":"Size Changing","age":"920"},
{"id":"1","power":"Flight","age":null}, {"id":"2","power":"Symbiote Costume","age":"457"}, {"id":"6","power":"Mind Control","age":"284"}, {"id":"7","power":null,"age":"104"},
{"id":"9","power":"Durability","age":"336"}]

```

The advantage of a dynamic endpoint is that by providing a variable in the URL, you are able to narrow the queried results to the value you exactly need from that dataset. In the following example, the *minimumId* variable is set to 5, so the "id" of the shown entries are bigger than 5:

```

<  →  ↻  Not Secure | mgergely-1.vpc.cloudera.com:18131/api/v1/query/5424/id/bigger/than/5?key=225342de-8588-4db9-8bfc-e52d9dccc34f
[{"id":"8","power":"Levitation","age":"659"}, {"id":"9","power":"Astral Trap","age":"204"}, {"id":"7","power":"Summoning","age":"681"}, {"id":"6","power":"Omnitrix","age":"758"}]

```

Using multiple variables in one endpoint

You can use multiple variables in a dynamic endpoint (*/id/between/{minimumId}/{maximumId}*), and you can also use the variables for different filters within a Materialized View query. In this case, the variables are replaced at every occasion by the same value you provide when using the dynamic endpoint.

Using SQL Stream Builder with Cloudera Data Visualization

You can create Business Intelligence reports from the Materialized Views created in SQL Stream Builder (SSB) using Cloudera Data Visualization. To integrate SSB with Data Visualization, you need to provide the PostgreSQL database information of SSB in Data Visualization.

About this task

After creating Materialized Views of your SQL Stream job, you can create visualized reports from your queried result with Cloudera Data Visualization.

Cloudera Data Visualization enables you to explore data and communicate insights by using visual objects. You can connect to your data in Cloudera Data Platform (CDP), create state-of-the-art visualizations on top of your datasets, build informative dashboards and applications, and publish them anywhere across the data lifecycle.

When connecting SSB with Cloudera Data Visualization, you need to provide the PostgreSQL database information that store the Materialized View data in the Cloudera Data Visualization web interface. For the connection, you can find the PostgreSQL database information in Cloudera Manager.

Before you begin

You have created a Materialized View query on your running SQL job.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click SQL Stream Builder from the list of services.
3. Click Configuration.
4. Filter down the configuration parameters to Materialized View Engine.

The list of Materialized View Engine parameters are displayed.

5. Save the ssb.mve.datasource related information.

Regarding the PostgreSQL information you need for the connection, the Database URL (JDBC) configuration parameter contains the PostgreSQL hostname, the PostgreSQL port and the PostgreSQL database name.

Filters (1) [Clear All](#)

SCOPE [Clear](#)

- SQL_STREAM_BUILDER-1 (Se... 32
- Load Balancer 43
- Materialized View Engine 57
- Streaming SQL Engine 83

CATEGORY

- Main 10
- Advanced 11
- Database 0

Database URL (JDBC)

ssb.mve.datasource.url
[ssb.mve.datasource.url](#)

Database User

ssb.mve.datasource.username
[ssb.mve.datasource.username](#)

Database Password

ssb.mve.datasource.password
[ssb.mve.datasource.password](#)

Materialized View Engine Default Group [Undo](#) ⓘ

jdbc:postgresql://docstest-1.docstest.root.hwx.site:5432/ssb_mve

Materialized View Engine Default Group [Undo](#) ⓘ

ssb_mve

Materialized View Engine Default Group ⓘ

.....

[Show All Descriptions](#)

When creating a data connection in Cloudera Data Visualization, you need to provide the connection information and credential as shown in the following example:

- Database Name: ssb_mve
- Database Host: docstest-1.docstest.root.hwx.site
- Database Port: 5432
- Database User: ssb_mve
- Database Password: [****POSTGRES*SQL DATABASE PASSWORD***]

The Database User and Database Password for the Materialized View Engine is configured based on what username and password was provided when installing the SQL Stream Builder service on your cluster.

What to do next

After collecting the necessary information to the connection, you need to access the Cloudera Data Visualization web interface and create a connection to the PostgreSQL database of SSB using the SQL Stream Builder connector. For more information, see the [Cloudera Data Visualization](#) documentation.



Important: The SQL Stream Builder connector is provided as a technical preview at this time in Cloudera Data Visualization. The tool is still under development and not recommended for a production environment.

Related Information

[Introduction to Materialized Views](#)

[Creating Materialized Views](#)