Cloudera DataFlow for Data Hub 7.2.15

Using Materialized Views in SQL Stream Builder

Date published: 2019-12-16
Date modified: 2022-05-12

https://docs.cloudera.com/

Legal Notice

Contents

Introduction to Materialized Views..4

Creating Materialized Views...4
Configuring Retention Time for Materialized Views...9

Using Dynamic Materialized View Endpoints...12

Configuring Materialized View database information.................................16

Using SQL Stream Builder with Cloudera Data Visualization........................18

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

- You need to have a running SQL job on which you apply the Materialized Views configuration.

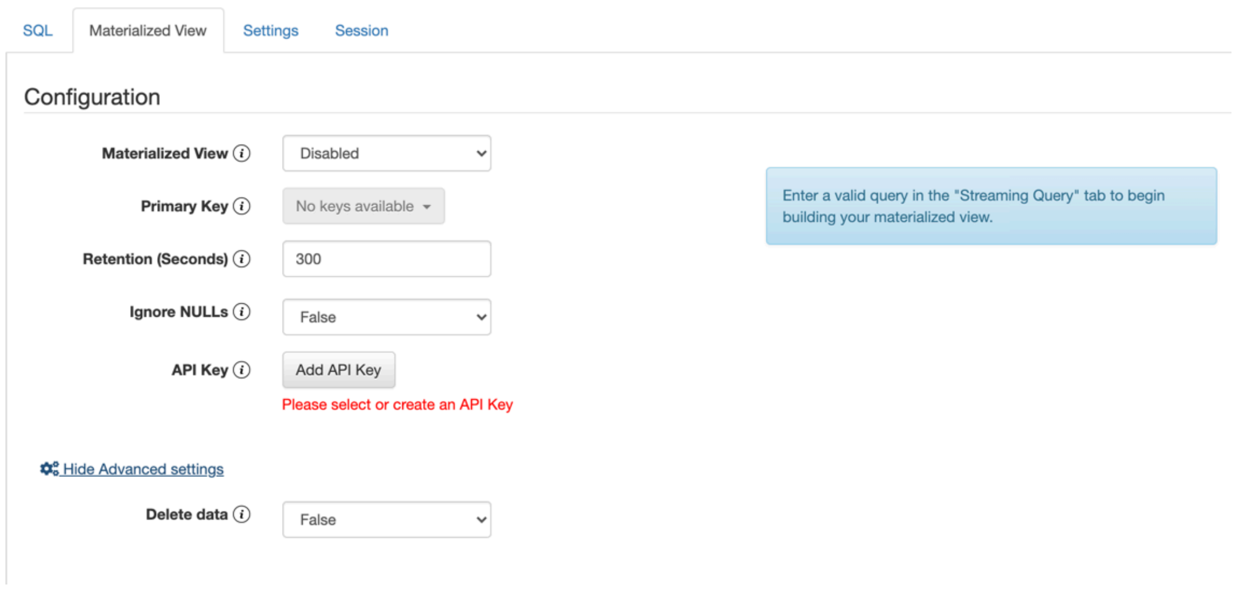
Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Run a SQL Stream job, and wait until data is shown on the results tab.

   For more information on running a SQL job, see the Running a SQL Stream job documentation.

3. Select the Materialized View sub-tab on the Compose tab.



Materialized View: Disabled
Primary Key: No keys available
Retention (Seconds): 300
Ignore NULLs: False
API Key: Add API Key
Please select or create an API Key
Delete data: False

Enter a valid query in the "Streaming Query" tab to begin building your materialized view.

4. Select Enabled for Materialized View.
5. Select a Primary Key.

   If this list is empty, then no SQL is specified on the SQL sub-tab, or that SQL 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.

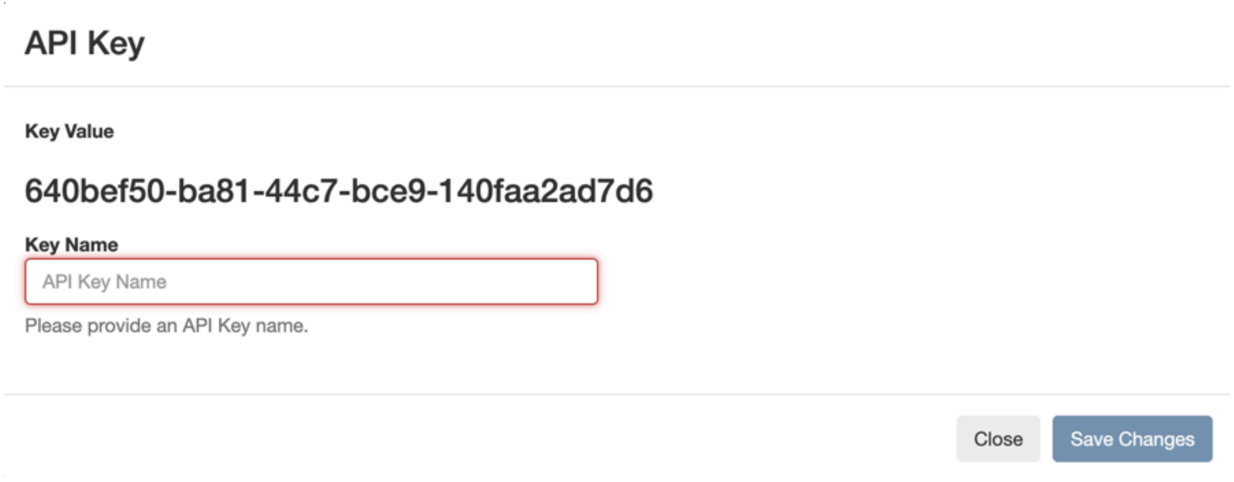
   Data not being mutated during this period is removed from the view.
7. Enable or disable Recreate on Job Start.

   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

640bef50-ba81-44c7-bce9-140faa2ad7d6

Key Name

API Key Name

Please provide an API Key name.

To check your created API keys:

a. Click Materialized Views from the main menu.

The Materialized Views page appears.

b. Click [show] under Key to see the API Key.

Note: You can delete the created API key by clicking the bin icon.

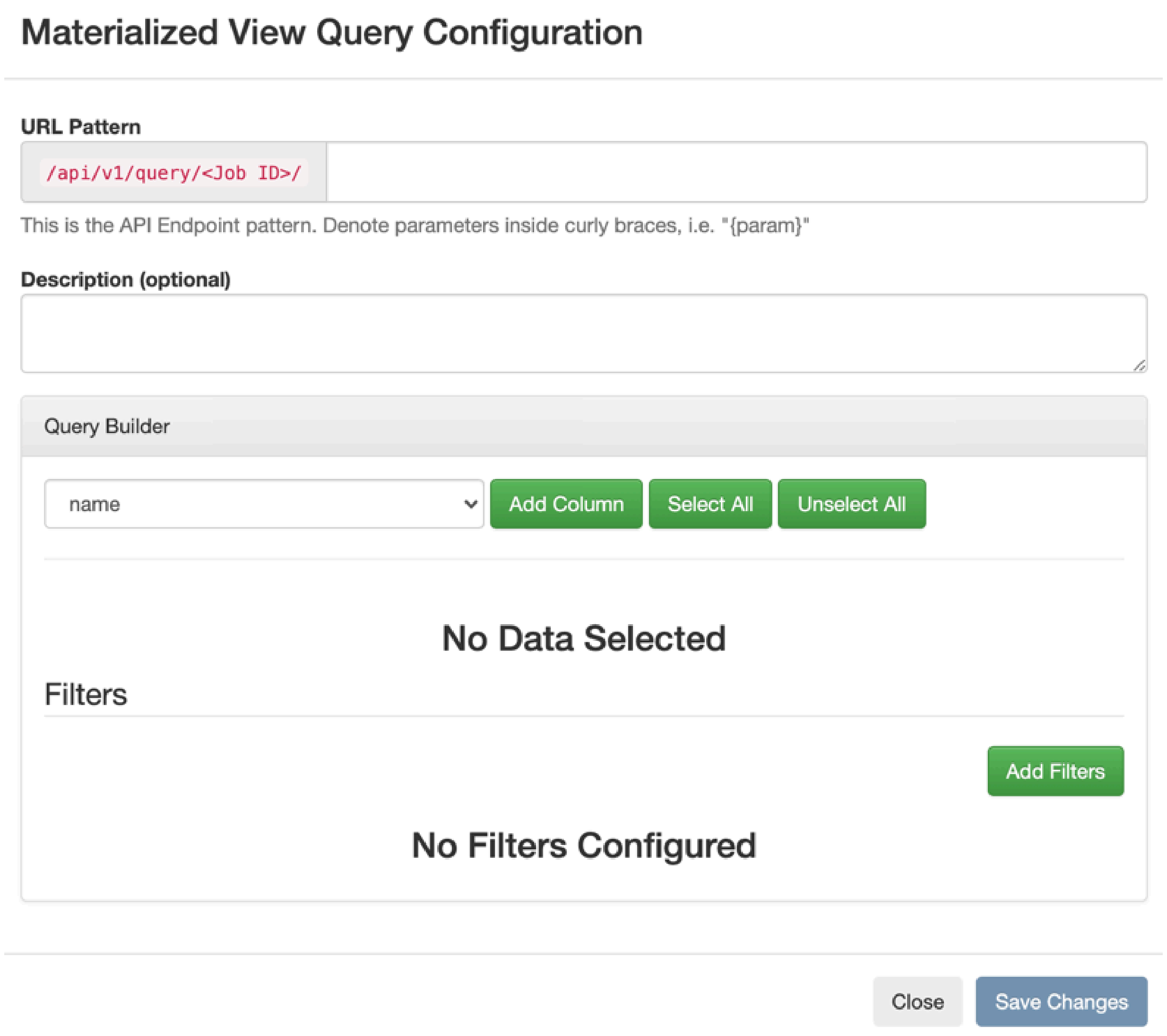
10. Click Advanced Settings for more configureation.

a) Set Delete data to *TRUE* to delete and recreate the Materialized View when restarting the SQL job. This setting is needed when the schema is changed for a job.

11. Click Apply Configuration.

12. Click Add Query to create the Materialized View query.

The Materialized View Query Configuration window appears.



Materialized View Query Configuration

URL Pattern

`/api/v1/query/<Job ID>/`

This is the API Endpoint pattern. Denote parameters inside curly braces, i.e. "{param}"

Description (optional)

Query Builder

name

No Data Selected

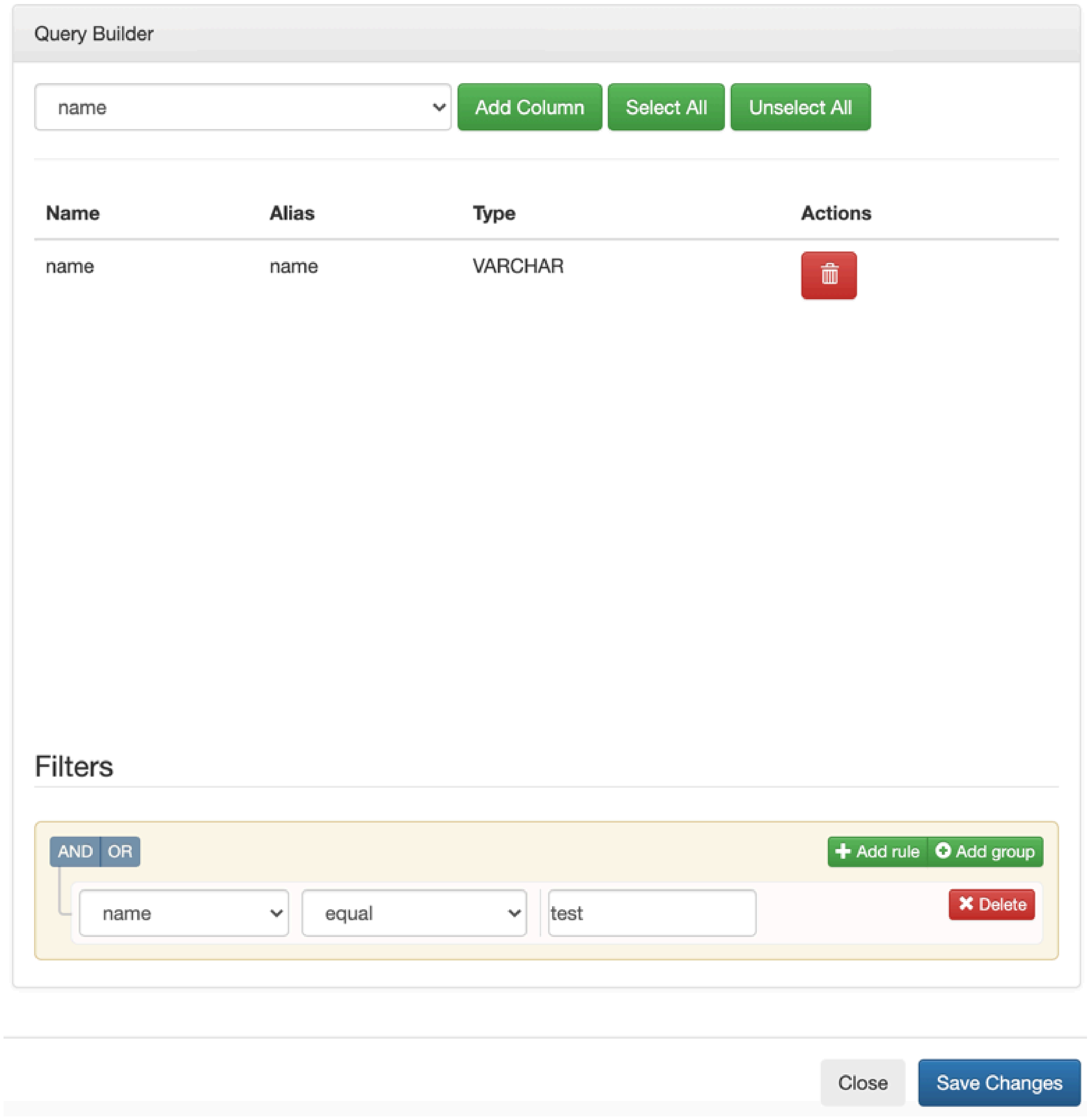
Filters

No Filters Configured

13. Provide a name to the URL Pattern.

14. Provide a description of the Materialized View Query, if needed.

15. Customize the Materialized View in the Query Builder.

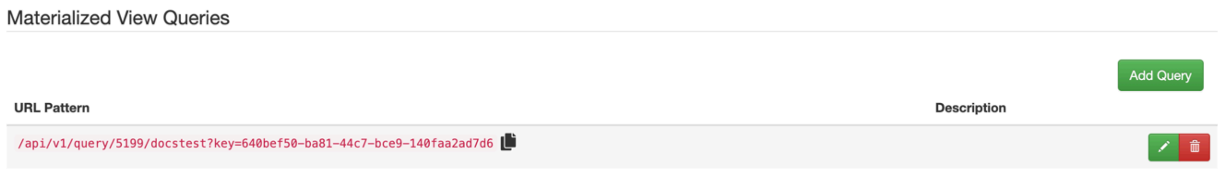


Name	Alias	Type	Actions
name	name	VARCHAR	

a) Select the columns of the SQL job you want to use in the Materialized View Query.
b) Click Add filters to apply computations and further enrichment of your data.
c) Click Add rule to save the filter.

16. Click Save Changes.

The Pattern URL appears under Materialized View Queries header.



URL Pattern	Description
/api/v1/query/5199/docstest?key=640bef50-ba81-44c7-bce9-140faa2ad7d6	

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.

Related Information

Running a SQL Stream job

Reviewing Materialized View database information

Using Dynamic Materialized View Endpoints

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

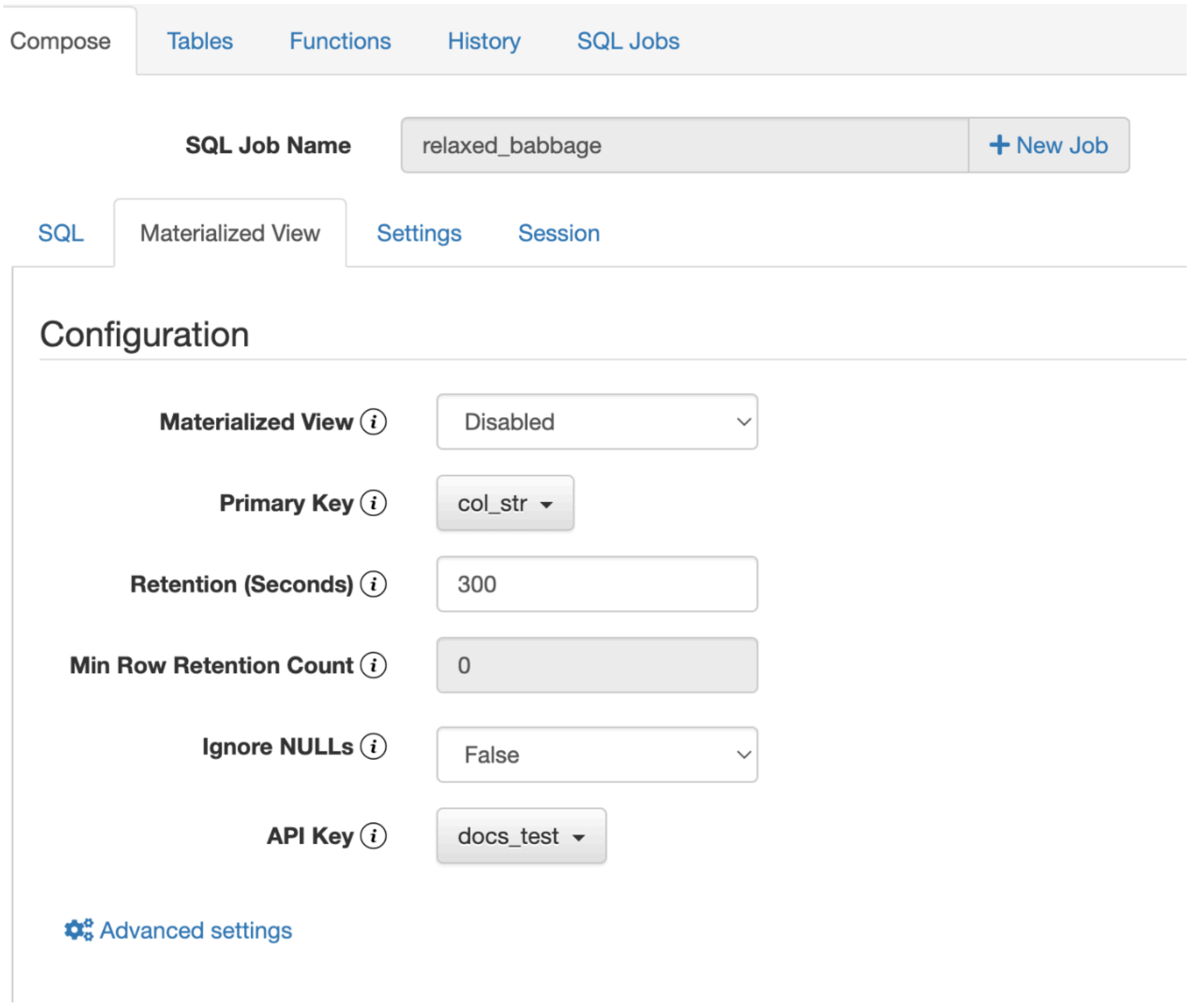
You can specify the Retention Time and Min Row Retention Count when creating a Materialized View for a SQL job on the Compose page of the Streaming SQL Console.

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.



SQL Job Name: relaxed_babbage

Configuration

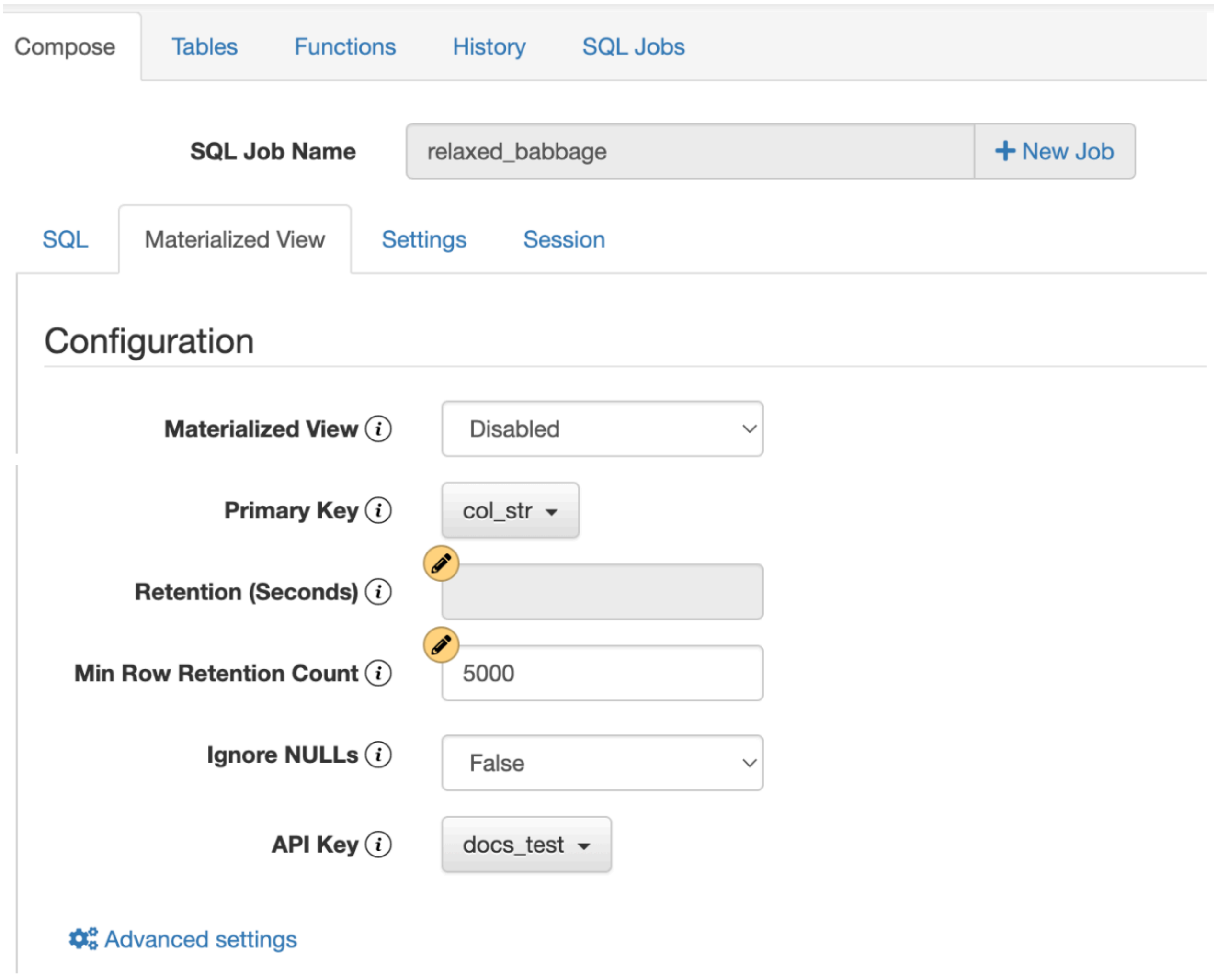
Materialized View	Disabled
Primary Key	col_str
Retention (Seconds)	300
Min Row Retention Count	0
Ignore NULLs	False
API Key	docs_test

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.



SQL Job Name: relaxed_babbage

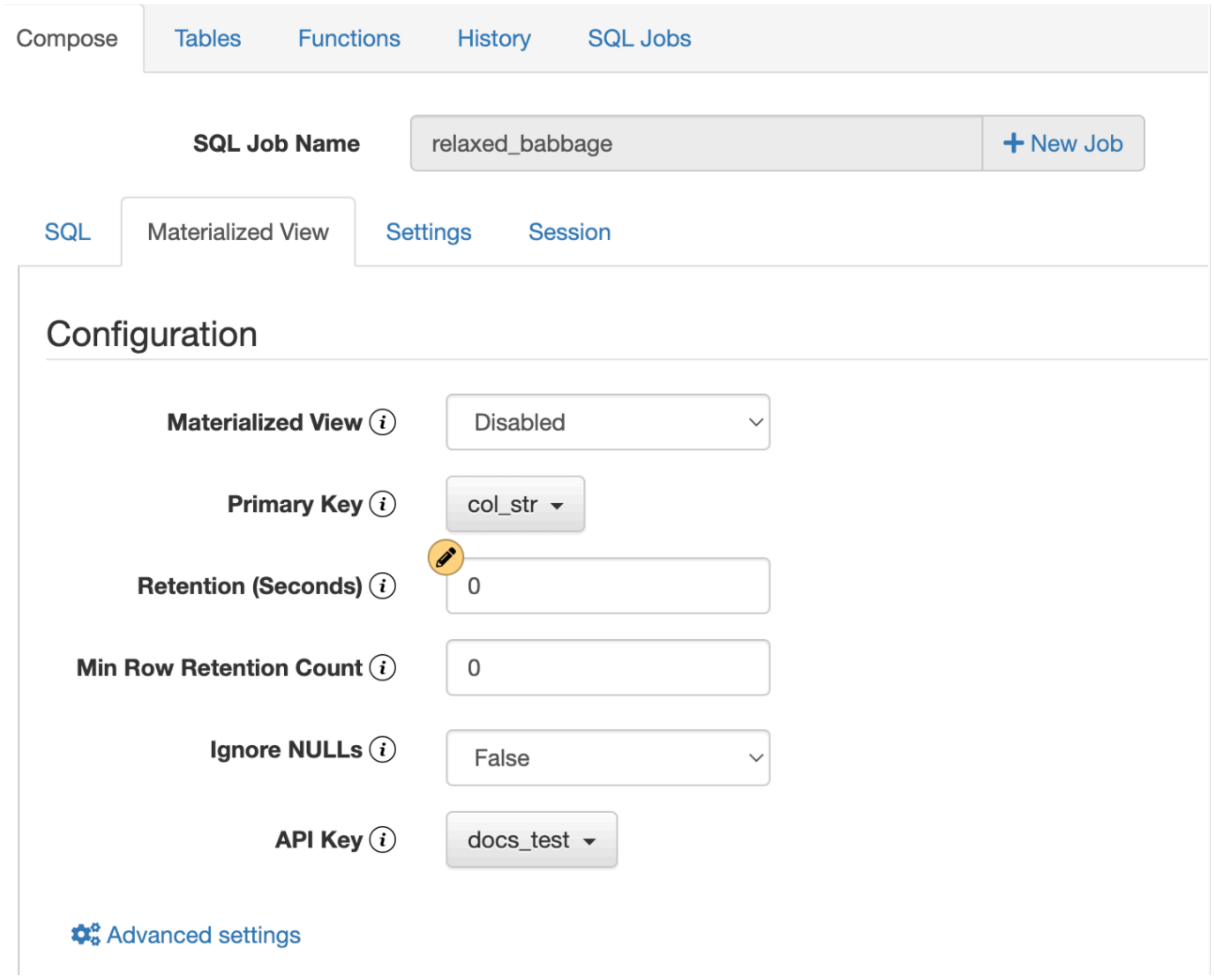
Configuration

Materialized View	Disabled
Primary Key	col_str
Retention (Seconds)	
Min Row Retention Count	5000
Ignore NULLs	False
API Key	docs_test

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:



SQL Job Name: relaxed_babbage

Configuration

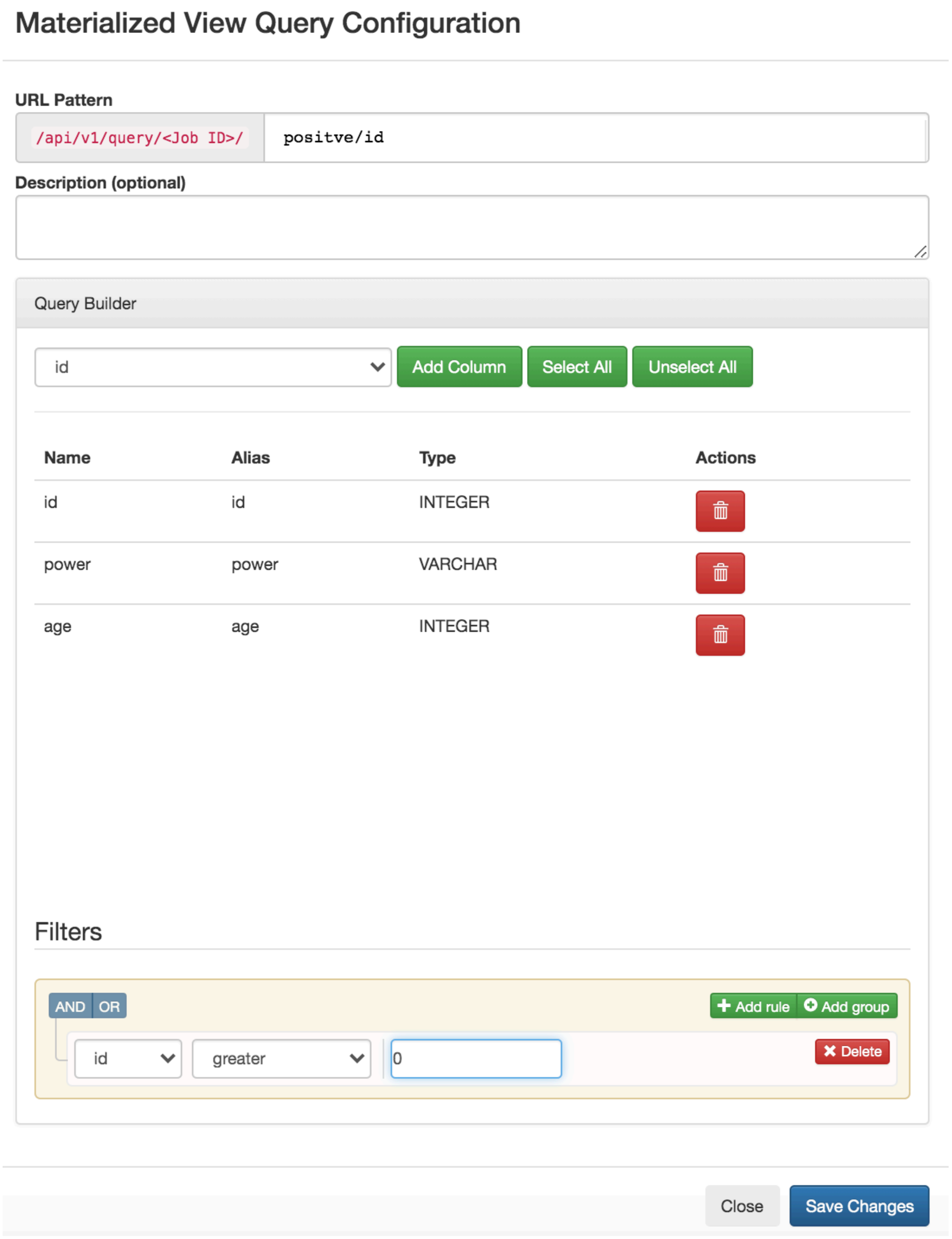
Materialized View	Disabled
Primary Key	col_str
Retention (Seconds)	0
Min Row Retention Count	0
Ignore NULLs	False
API Key	docs_test

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*:


Materialized View Query Configuration

URL Pattern

/api/v1/query/<Job ID>/ positve/id

Description (optional)

Query Builder

id

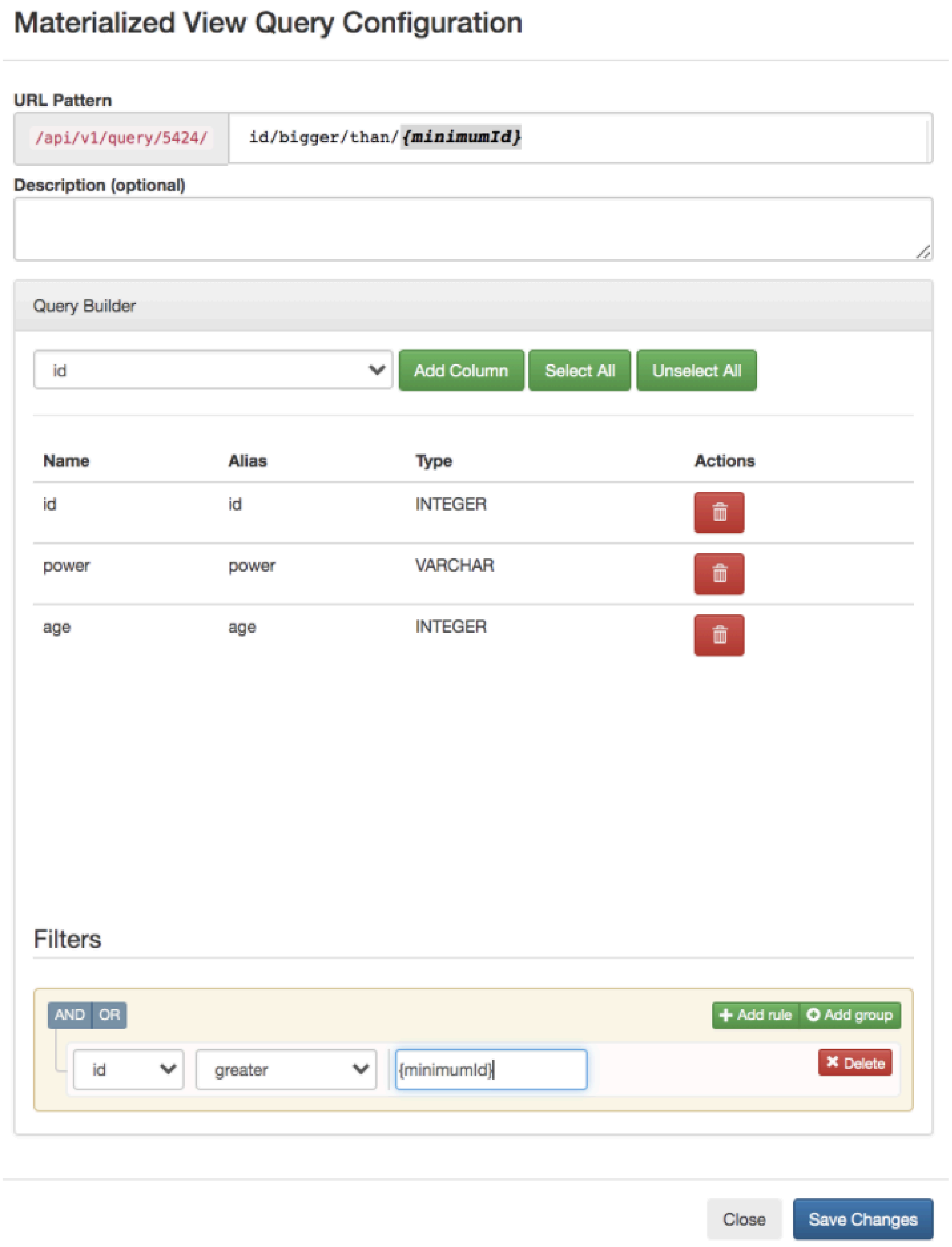
Name	Alias	Type
id	id	INTEGER
power	power	VARCHAR
age	age	INTEGER

Filters

AND OR

id greater 0

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}*:



Materialized View Query Configuration

URL Pattern

`/api/v1/query/5424/` `id/bigger/than/{minimumId}`

Description (optional)

Query Builder

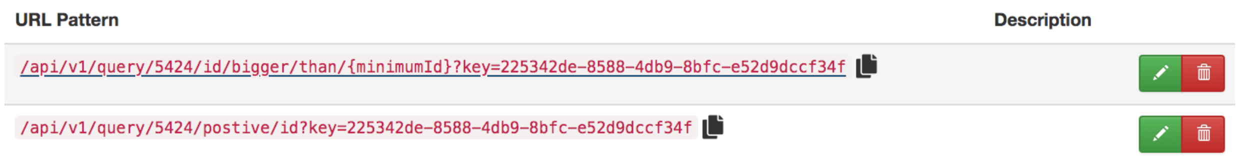
Name	Alias	Type	Actions
id	id	INTEGER	
power	power	VARCHAR	
age	age	INTEGER	

Filters

id greater {minimumId}

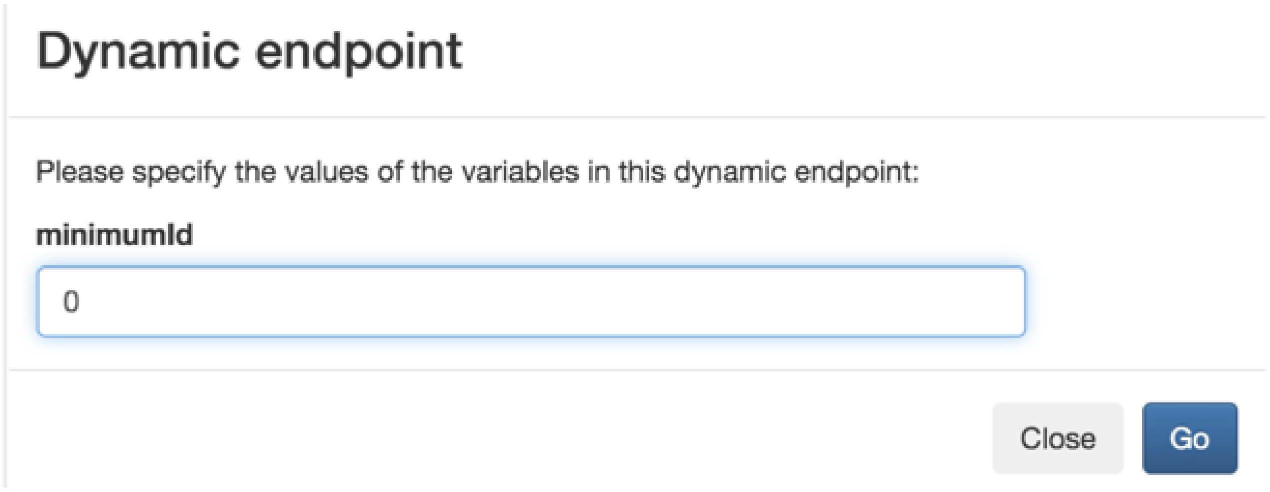
Using dynamic endpoints

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:



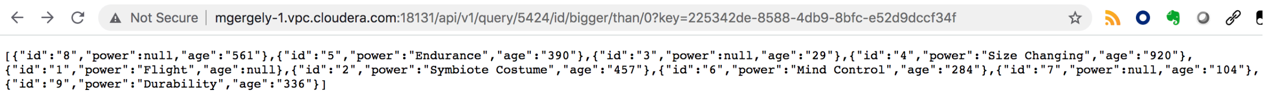
Dynamic endpoint

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

minimumId

0

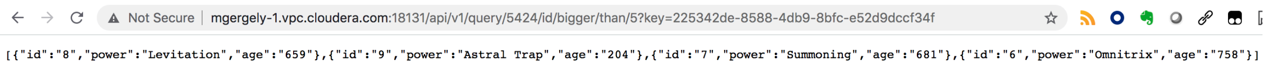
After entering a value, the filtered results are shown in a new browser tab:



mgergely-1.vpc.cloudera.com:18131/api/v1/query/5424/id/bigger/than/0?key=225342de-8588-4db9-8bfc-e52d9dccf34f

```
[{"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:



mgergely-1.vpc.cloudera.com:18131/api/v1/query/5424/id/bigger/than/5?key=225342de-8588-4db9-8bfc-e52d9dccf34f

```
[{"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.

Configuring Materialized View database information

In CDP Public Cloud, PostgreSQL is automatically configured for SQL Stream Builder (SSB) when creating your Data Hub cluster using the Streaming Analytics cluster template. You can review the database configuration for SSB using Cloudera Manager.

About this task

SSB uses PostgreSQL database to store the queried data of the Materialized View feature. In CDP Public Cloud, you can only use PostgreSQL as a database for SSB. In case you need to connect to the PostgreSQL database of SSB, you can review the necessary information of the database in Cloudera Manager. You can also configure the user and the password of the database using CLI, and then updating the existing information in Cloudera Manager.

Where to find the PostgreSQL database information?

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the list of services.
3. Select Clusters SQL Stream Builder .
4. Click Configuration.
5. Filter down the configuration parameters to Materialized View Engine.

   The following information is listed about the PostgreSQL database:

   - Database Name
   - Database Host
   - Database Port
   - Database User
   - Database Password

When you want to change the default configuration of PostgreSQL database, first you need to change the information in PostgreSQL and then update the configuration in Cloudera Manager for SSB.

Procedure

1. Connect to the database host using ssh.

```
ssh [***WORKLOAD USERNAME***]@[***DATABASE HOST***]
Password: [***YOUR WORKLOAD PASSWORD***]
```

2. Connect to PostgreSQL.

```
sudo -U postgres psql
```

   Note: As an EnvironmentUser, you will be prompted to provide the password related to the PostgreSQL client. In this case, please reach out to your administrator to provide the correct password for the PostgrSQL client connection.

3. Change the username or the password for the database.

```
ALTER USER eventador_admin WITH PASSWORD [***NEW PASSWORD***];
ALTER USER eventador_admin RENAME TO [***NEW USERNAME***];
```

   Note: When changing the database username, the existing password for the username is deleted. You must change the password as well when renaming the user of the database.

   After changing the password or username, you need to update the database information in Cloudera Manager.

4. Navigate to Management Console Environments , and select the environment where you have created your cluster.
5. Select Cloudera Manager from the list of services.
6. Select Clusters SQL Stream Builder .
7. Click Configuration.
8. Filter down the configuration parameters to Materialized View Engine.

   The list of Materialized View Engine paramaters are displayed.

9. Update the Database user or the Database password field based on the new parameters you provided in PostgreSQL.

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.

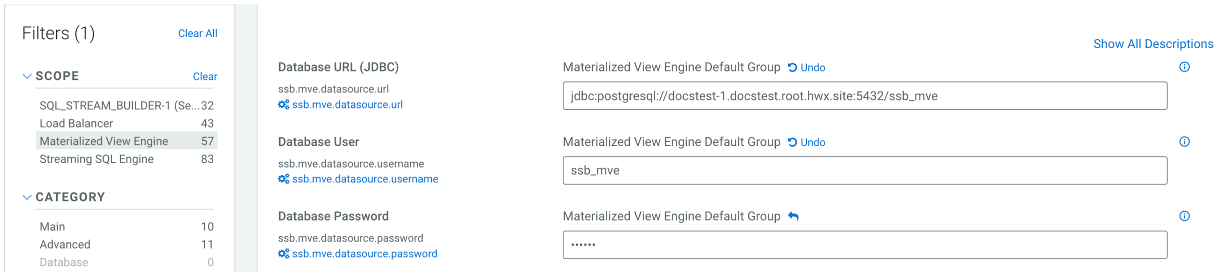
Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select Cloudera Manager from the list of services.
3. Select Clusters SQL Stream Builder .
4. Click Configuration.
5. Filter down the configuration parameters to Materialized View Engine.

   The list of Materialized View Engine paramaters are displayed.

6. 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

Show All Descriptions

Database URL (JDBC)
ssb.mve.datasource.url
ssb.mve.datasource.url
Materialized View Engine Default Group Undo
jdbc:postgresql://docstest-1.docstest.root.hwx.site:5432/ssb_mve

Database User
ssb.mve.datasource.username
ssb.mve.datasource.username
Materialized View Engine Default Group Undo
ssb_mve

Database Password
ssb.mve.datasource.password
ssb.mve.datasource.password
Materialized View Engine Default Group
••••••

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: *[***POSTGRESQL DATABASE PASSWORD***]*

The Database User and Database Password for the Materialized View Engine is configured automatically when creating your Data Hub cluster. In case you cannot access the Database information, the Administrator of the environment can help to change the configuration as described in Configuring Materialized View database information.

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

Reviewing Materialized View database information