On this page

Row-Access & Column-Masking Policies

Row-access and column-masking policies may be applied to tables, views, and columns via user-defined functions (UDFs). Using these policies, you can control access to sensitive data based upon the rules and conditions you need to maintain compliance or adhere to regulatory requirements, while also removing the need to produce a secondary set of data with protected information manually removed.

> **Note:**
> Row-access and column-masking policies are supported on all object storage, metastore, and database sources supported in Dremio Cloud (see Connecting to your Data), as well as Dremio Arctic (Preview) (see Dremio Arctic).

The following restrictions apply to policies and UDFs:

- Row-access and column-masking policies are not branch-aware when working with Dremio Arctic sources.
- Only users with the `ADMIN` role can create UDFs.
- UDFs can only have one owner, which is the user that created the UDF, by default.
- You can transfer ownership of a UDF using the `GRANT OWNERSHIP` command (see Privileges).
- The owner of a UDF that serves as a policy must have the `EXECUTE` privilege for that UDF.

Column-Masking

Column-masking is a way to mask-–or scramble–-private data at the column-level dynamically prior to query execution. For example, the owner of a table or view may apply a policy to a column to only display the year of a date or the last four digits of a credit card.

Column-masking policies may be any UDF with a scalar return type that is identical to the data type of the column on which it is applied. However, only one column-masking policy may be applied to each column.

In the following example of a user-defined function, only users within in the Accounting department in the state of California (CA) may see an entry's social security number (ssn) if the record lists an income above $10,000, otherwise the SSN value is masked with XXX-XX-.

Column-masking policy example

Row-Access

Row-access policies are a way to control which records in a table or view are returned for specific users and roles. For example, the owner of a table or view may apply a policy that filters out customers from a specific country unless the user running the query has a specific role.

Row-access policy example

```
CREATE FUNCTION country_filter (country VARCHAR) RETURNS BOOLEAN RETURN SELEC
```

Row-access policies may be any boolean UDF applied to the table or view. The return value of the UDF is treated logically in a query as an AND operator included in a WHERE clause. The return type of the UDF must be BOOLEAN, otherwise Dremio will give an error at execution time.

User-Defined Functions

A user-defined function, or UDF, is a callable routine that accepts parameters and returns a scalar value.

The UDFs which serve as the basis for filtering and masking policies must be defined independently of your sources. Not only does this allow organizations to use a single policy for multiple tables and views, but this also restricts user access to policies and prevents unauthorized tampering. Modifying a single UDF automatically updates the policy in the context of any tables or views using that access or mask policy.

The following process describes how policies are enforced with Dremio:

1. A user with the ADMIN role creates a UDF to serve as a security policy.
2. The administrator then sets the security policy to one or more tables, views, and/or columns.
3. Dremio enforces the policy at runtime when an end-user performs a query.

Creating UDFs and attaching security policies is done through SQL commands. Policies are applied prior to execution during the query planning phase. At this point, Dremio checks first the table/view for a row-access policy and then each column accessed for a column-masking policy. If any policies are found, they are automatically applied to the policy's scope using the associated UDF in the query plan.

Query Substitutions

Row-access and column-masking function act as an “implicit view,” replacing a table/view reference in an SQL statement prior to processing the query. This implicit view is created through an examination of each policy applied to a table, view, or column.

For example, jdoe@dremio.com has SELECT access to table_1. However, the column-masking policy protect_ssn is set for the column_1 column with a UDF to replace all but the last four digits of a social security number with X for anyone that is not a member of the Accounting department, or this user. When they run a query in Dremio that includes this column-masking policy, the following occurs:

1. During the SQL Planning phase, Dremio identifies which tables, views, and columns are being accessed (table_1) and whether security policies must be enforced.
2. The engine searches for any security policies set to the associated objects, such as protect_ssn (see Examples of UDFs below).
3. When the protect_ssn policy is found for the object affected by the query, the query planner immediately modifies the execution path to incorporate the masking function.
4. Query execution proceeds as normal with the associated UDF included within the execution path.

To view all existing UDFs created in Dremio, use the SHOW FUNCTIONS SQL command.

Listing Existing Policies

To view row-access and column-masking policies, use a SELECT statement with the target table/view, system table, and policies specified.

List existing column-masking and row-access policies

```
SELECT view_name, masking_policies, row_access_policies FROM
sys.project.views;
SELECT table_name, masking_policies, row_access_policies FROM
sys.project."tables";
```

To view all column-masking policies set for a given table, use the DESCRIBE TABLE command.

Setting a Policy

To create a row-access or column-masking policy, you must perform the following steps using the associated SQL commands:

1. Create a new UDF or replace an existing one using the `CREATE \[OR REPLACE\]` function command.

   Create or replace UDF

   ```
   CREATE FUNCTION country_filter (country VARCHAR) RETURNS BOOLEAN RETURN
   ```

2. Grant the EXECUTE privilege to the owner of the UDF.

   Grant EXECUTE privilege

   ```
   GRANT EXECUTE ON FUNCTION country_filter TO user 'UDF_owner@dremio.com';
   ```

3. Create a policy to apply the function using either ADD ROW ACCESS POLICY for row-level access or SET MASKING POLICY for column-masking. These may be used with the CREATE TABLE, CREATE VIEW, ALTER TABLE, and ALTER VIEW commands.

```
-- Add row-access policy ALTER TABLE e.employee ADD ROW ACCESS POLICY co
```

Note:
Both row-access and column-masking UDFs may be applied in a single security policy, or set individually.

Dropping a Policy

To remove a security policy from a table, view, or row, use the UNSET MASKING POLICY or DROP ROW ACCESS POLICY syntax with ALTER TABLE/VIEW.

Remove security policy

```
ALTER TABLE w.employee DROP ROW ACCESS POLICY country_filter(country); ALTER
```

Examples of UDFs

The following are examples of user-defined functions that you may create with Dremio.

Column-Masking

Redact SSN

```
CREATE FUNCTION protect_ssn (val VARCHAR) RETURNS VARCHAR RETURN SELECT CASE
```

Use column-masking and row-access policies

```
CREATE FUNCTION lower_country(country VARCHAR) RETURNS VARCHAR RETURN SELECT
```

Use STRUCT

```
-- CREATE TABLE struct_demo (emp_info struct <name : VARCHAR>); INSERT INTO r
```

Use LIST

Row-Access

Use simple filter expressions

```
CREATE FUNCTION country_filter (country VARCHAR) RETURNS BOOLEAN RETURN SELEC
```

Match users

```
CREATE FUNCTION query_1(my_value varchar) RETURNS BOOLEAN RETURN SELECT CASE
```

Was this page helpful? Yes No