dremio Documentation Dremio Cloud

On this page

ALTER TABLE

Update a table's definition or schema.

Syntax

```
-- Add Columns ALTER TABLE <table_name> ADD COLUMNS ( <column_name1> <data_ty
```

Parameters

<table_name> String

The name of the table that you want to alter. The source can be in the scratch directory or a data lake source.

ADD COLUMNS (<column_name1> <data_type>, <column_name2> <data_type>, ...) String

Creates one or more columns that have the specified names, data types, and character limits. The size is an attribute of the data type.

DROP COLUMN <column_name> String

Drops the specified column. This action cannot be undone.

{ ALTER | MODIFY } COLUMN <old_column_name> <new_column_name> <data_type>

String

Changes the data type or name for a specified column. If you are only changing the name, leave the data type in the clause. The size is an attribute of the data type.

Note:
You can only rename columns of scratch and Iceberg tables.

REFRESH METADATA

refresh will enable you to only update specified partitions. Optional clauses are available for refreshing a table's metadata. If you choose to use two or more of these clauses, they must be entered in a specified priority order:

1. Promotion option: either `AVOID PROMOTION` or `AUTO PROMOTION`
2. Update option: either `FORCE UPDATE` or `LAZY UPDATE`
3. Missing option: either `MAINTAIN WHEN MISSING` or `DELETE WHEN MISSING`

Optional

Note:
If the table is not partitioned, you will receive an error when you attempt to refresh the metadata using the `FOR PARTITIONS` clause.

FOR PARTITIONS (<partition_name> = '<value>') String

Use this clause to do a partial refresh of the table's metadata.

Optional

- The `<partition_name>` identifies the name of the partition to be refreshed.
- The `<value>` identifies the specific partition that should be refreshed. Must be contained in single quotes.

{ AVOID I AUTO } PROMOTION

Clauses that determine whether files and folders are promoted to datasets when you run a query.

Optional

The `AVOID PROMOTION` prevents queries from promoting files/folders to datasets. The `AUTO PROMOTION` allows queries to promote files/folders to datasets. This is the default option when you do not include a promotion clause.

{ FORCE I LAZY } UPDATE

Optional

Clauses that determine whether metadata is updated when you run a query.

The `FORCE UPDATE` forces a full update of metadata. The `LAZY UPDATE` does *not* perform a full update of metadata. This is the default option when you do not include an update clause.

{ MAINTAIN I DELETE } WHEN MISSING

Clauses that determine how missing metadata is handled when you run a query.

dremio
Documentation

Dremio Cloud

The `MAINTAIN WHEN MISSING` Prevents missing metadata from being deleted during refresh. `DELETE WHEN MISSING` deletes missing metadata during refresh. This is the default option when you do not include a clause.

FORGET METADATA

Deletes the metadata information stored in Dremio for the specified table until the next metadata refresh. The dataset can still be queried using SQL.

<reflection_name> String

The name to give to the new reflection.

DIMENSIONS (<column_name1>, <column_name2>, ...) String

The columns to include as dimensions in the reflection.

MEASURES (<column_name1> (<aggregation_type>), <column_name2> (<aggregation_type

String

The columns to include as measures in the reflection, and the type of aggregation to perform on them. The possible types are COUNT, MIN, MAX, SUM, and APPROXIMATE COUNT DISTINCT.

PARTITION BY (<column_name1>, <column_name2>, ...) String

The columns on which to partition the data horizontally in the reflection.

LOCALSORT BY (<column_name1>, <column_name2>, ...) String

The columns on which to sort the data that is in the reflection.

ARROW CACHE

Specifies that you want Dremio to convert data from your reflection's Parquet files to the Apache Arrow format when copying that data to executor nodes. Normally, Dremio copies data as-is from the Parquet files as-is to caches on executor nodes, which are nodes that carry out the query plans devised by the query optimizer. Enabling this option can improve query performance even more. However, data in the Apache Arrow format requires more space on the executor nodes than data in the default format. You can use this option with Amazon S3.

DISPLAY (<column_name1>, <column_name2>, ...) String

The columns to include in the reflection.

{ DEFAULT ENGINE | ENGINE { <engine_name> | <engine_uuid> } } String

Specify an engine to route reflections to either by the name or UUID of the engine. If not specified, the default engine will be used.

{ ALTER | MODIFY | CHANGE } COLUMN (<source_col_name> <new_col_name> <data_type>)

String

Only three types of changes to primitive types are allowed:

- int to long
- float to double
- decimal(P, S) to decimal(P', S), if you are widening the precision

You can alter columns that use complex types by using either of these two sets of syntax:

Set 1

- struct_type: `ROW( name primitive_or_complex_type, .. )`
- list_type: `ARRAY(primitive_or_complex_type)`

Examples:

```
ROW(innerfield INT, anotherinnerfield DOUBLE)
ARRAY(INT)
ROW(innerfield ARRAY(INT))
ARRAY(ROW(innerfield INT))
```

Set 2

- struct_type: `STRUCT <name : primitive_or_complex_type, ...>`
- list_type: `{ LIST | ARRAY } < primitive_or_complex_type >`

Examples:

```
STRUCT<innerfield : INT, anotherinnerfield : DOUBLE>
LIST<INT>
ARRAY<INT>
STRUCT<innerfield : LIST<INT>>
LIST<STRUCT<innerfield : INT>>
```

ADD { COLUMN I COLUMNS } (<column_name1> <data_type> [, <column_name2> <data_type> ...]) [BEFORE <column_name>]

String

Appends one or more columns that have the specified names and data types.

These are the supported primitive types:

- DATE
- FLOAT
- DECIMAL
- DOUBLE
- INTERVAL
- INT
- BIGINT
- TIME
- TIMESTAMP
- VARCHAR (The length is always 65536 bytes. If a length is specified, it is ignored.)

You can define complex types by using either of these two sets of syntax:

Set 1

- struct_type: `ROW( name primitive_or_complex_type, .. )`
- list_type: `ARRAY(primitive_or_complex_type)`

Examples:

```
ROW(innerfield INT, anotherinnerfield DOUBLE)
ARRAY(INT)
ROW(innerfield ARRAY(INT))
ARRAY(ROW(innerfield INT))
```

Set 2

- struct_type: `STRUCT <name : primitive_or_complex_type, ... >`
- list_type: `{ LIST | ARRAY } < primitive_or_complex_type >`

Examples:

```
STRUCT<innerfield : INT, anotherinnerfield : DOUBLE>
LIST<INT>
ARRAY<INT>
STRUCT<innerfield : LIST<INT>>
LIST<STRUCT<innerfield : INT>>
```

{ ADD | DROP } PARTITION FIELD { <column_name> | <partition_transform> } String transformation functions. DROP drops the partition definition.

These are the partition-transformation functions:

Transform	Description
identity(<col>)	Explicitly specified identity transform
year(<col>)	Partition by year. The column must use the TIMESTAMP data type.
month(<ts_col>)	Partition by month. The column must use the TIMESTAMP data type.
day(<ts_col>)	Partition by day. The column must use the TIMESTAMP data type.
hour(<ts_col>)	Partition by hour. The column must use the TIMESTAMP data type.
bucket(<count>, <col>)	Partition by hashed value into <count> buckets
truncate(<length>, <col>)	Partition by truncated value. • Strings are truncated to the specified length. • Integer and biginteger values are truncated to bins. Example: truncate(10, i) produces 0, 10, 20, and so on.

COLUMN String

An SQL keyword to indicate that the target of the RENAME action is a column.

<old_column_name> String

The current name of the column.

<new_column_name> String

The new name to give to the column.

MODIFY COLUMN <column_name> String

Specifies the column to which the masking policy will apply and mask data for. The UDF serving as the masking policy must accept and return the same data type as the column it is masking.

Specifies the function to use with this security policy. If a function with this name does not exist, then the affected table/view will not be reachable until the policy is dropped or a UDF created.

Examples

Add a column

```
ALTER TABLE services ADD COLUMNS (county varchar)
```

Modify a column

```
ALTER TABLE services MODIFY COLUMN tip_amount tip_amount DECIMAL
```

Refresh all the metadata for a table

```
ALTER TABLE services REFRESH METADATA
```

Refresh all the metadata for a table using optional clauses

```
ALTER TABLE services REFRESH METADATA AUTO PROMOTION LAZY UPDATE MAINTAIN WHE
```

Refresh the metadata for a single partition

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" REFRESH METADATA FOR PAR
```

Refresh the metadata for a single partition using optional clauses

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" REFRESH METADATA FOR PAR
```

Forget the metadata for a table

Create a raw reflection that sorts customers by last name and partitions them by country

```
ALTER TABLE "@user1"."customers" CREATE RAW REFLECTION customers_by_country U
```

Create an aggregate reflection that counts the cities per state and sorts by state

```
ALTER TABLE Samples."samples.dremio.com"."zips.json" CREATE AGGREGATE REFLECT
```

Routing Reflections

```
ALTER TABLE "Table 1" ROUTE REFLECTIONS TO ENGINE "Engine 1" ALTER TABLE "Vie
```

Adding a column for an Apache Iceberg Table

```
ALTER TABLE myTable ADD COLUMN (address VARCHAR)
```

Changing the data type of a column to BIGINT for an Apache Iceberg Table

```
ALTER TABLE myTable ALTER COLUMN id id BIGINT
```

Set a column-masking policy that takes multiple columns

```
ALTER TABLE e.employees MODIFY COLUMN ssn_col SET MASKING POLICY protect_ssn
```

Unset a column-masking policy

```
ALTER TABLE e.employees MODIFY COLUMN ssn_col UNSET MASKING POLICY protect_ss
```

Add a row-access policy to a table

Remove a row-access policy from a table

```
ALTER TABLE employees MODIFY COLUMN ssn_col DROP ROW ACCESS POLICY protect_ss
```