ALTASQL FOR SNOWFLAKE

Relational Metadata Management System (RMMS)
Administration and Governance Guide

Contents

Overview

AltaSQL is not an application. It is a meta-data database implemented as an SQL engine layered over the Snowflake database. It is not a Snowflake "Native Application"; it is a SHAREd database using SECURE User Defined Functions (UDFs) to deliver its capabilities. We refer to the AltaSQL UDFs with the abbreviation "ASDF".

The use of Secure UDFs ensures the integrity and security of your data. Nothing is installed in your Snowflake environment. AltaSQL has no access to your data.

AltaSQL both requires and enables the same types of administration and governance as Snowflake. Administration and governance are implemented using Snowflake constructs. In conjunction with AltaSQL features, AltaSQL offers new capabilities for administration and governance.

AltaSQL is designed to work with your existing and future development and deployment processes.

There are very few pre-defined processes. Though we offer recommendations, you should expect to tailor AltaSQL to your organization's needs.

Being pure SQL, with the underlying foundation consisting of Snowflake objects, AltaSQL is also designed for changes and evolution.

AltaSQL Installation

There is no AltaSQL code base to install. AltaSQL is delivered via a Snowflake SHAREd database. All functionality is delivered via SECURE UDFs from a schema in the shared database.

The shared database also includes a schema with sample meta-data repositories.

Use:
`CREATE DATABASE <name> FROM SHARE <provider_account>.<share_name>`
for creating a database from an `INBOUND SHARE`.

We refer to the created database as `ASQL_DB`, although the name you choose may be different. The schema with the functions is named `ASDF_SCHEMA`, and the schema with the sample meta-data repositories as `SBLDR_META_REPO_SCHEMA`.

Recommended AltaSQL Access

AltaSQL is intended to provide self-service capabilities to the widest possible user base in your organization.

AltaSQL's main function is generating SQL statements from meta-data repositories. ***The user is responsible for executing the generated code; we call this "Look before you execute."*** This

enables you to examine and modify the generated code if necessary. This also enables its use with dataops and data transformation pipelines.

After the desired database is created from the `INBOUND SHARE`, access must be granted to `ASQL_DB`, and its schemas.

Granting Access to SHARE

We recommend creating a specific role for `GRANT IMPORTED PRIVILEGES ON DATABASE` to <role> and then granting that role to `PUBLIC`.

If your policies prevent granting access directly to Public, we recommend granting access to the widest possible audience. For users to access and use repositories, the role they use must both be able to execute ASDFs and perform SELECT on the desired repositories. The ASDF usage role should be granted to the SELECT roles. This technique is handled by the script *CREATE_META_DATA_REPO_DB* which creates a database for managing and governing repositories. The script is available for download from the AltaSQL.io website Products menu.

We recommend that a dedicated Snowflake XSM warehouse be granted to the roles used for wide access. AltaSQL requires very little CPU, but generated DML should be executed separately from the wide access role. The *CREATE_META_DATA_REPO_DB* script creates a named XSM warehouse based on the database name. Depending on usage, you can alternatively create an XSM warehouse to be used by many or all AltaSQL users.

Access and Security

1. AltaSQL SQL functions require access to a meta-data repository by setting the session variable `SBLDR_META_REPOSITORY` to the desired meta-data context. The meta-data context will typically be either a physical meta-data repository or a VIEW created from multiple meta-data repositories. However, any construct that supports the repository columns and can be accessed by `IDENTIFIER($SBLDR_META_REPOSITORY)` can be used.
2. Executing AltaSQL SQL functions only generates `SQL Statement`; they do not execute generated `SQL`.
3. ***Users can only execute generated SQL statements if they have appropriate privileges; AltaSQL cannot override your security settings.***
4. Users can only access meta-data repositories if they have appropriate Snowflake `USAGE` and `SELECT` privileges on the meta-data repository specified by meta-data context. Users can only create and manage repositories if they have appropriate privileges.
5. AltaSQL simplifies usage in Snowsight and standard SQL tools by minimizing the number of worksheets needed. With typical usage, one worksheet's context is used for executing ASDFs using an extra-small Warehouse and another for executing the generated SQL. The second worksheet typically has appropriate ROLE and Warehouse settings.

a. With bulk generation, the user can create an entire schema. Further, the user can create all the desired objects as `TEMPORARY`. (See: Bulk Operations in the Technical Reference.)
b. Row access policies can be applied to meta-data repositories, including referencing AltaSQL Tags.

Ready-To-Execute AltaSQL SDFs Excel Spreadsheet

To facilitate productivity and ease of use, we provide an Excel spreadsheet with Ready-to-Execute AltaSQL defined function calls for the user to copy, paste and modify i.e.,

```
SELECT
      ASQL_DB.ASDF_SCHEMA.SBLD_REPO_FROM_INFO_SCHEMA(
            META_REPO_NAME => <required>,
            SOURCE_DB => <required>,
            SOURCE_SCHEMA => 'NULL',
            TO_SCHEMA => 'NULL',
            TO_DB => 'NULL',
            VIEW_PREFIX => 'NULL',
            VIEW_SUFFIX => 'NULL',
            TABLE_CONTROL => 'NULL',
            DATA_RETENTION_TIME_IN_DAYS => NULL,
            OBJECT_TAG => 'NULL',
            ROW_ACCESS => 'NULL') ;
```

`<required>` arguments must be supplied and are used even when `SBLD_IS_NULL`. Optional arguments are preset to quoted `'NULL'` or true NULL.

Simply changing the argument values eliminates the need to rewrite the function call for different results, saving keystrokes.

Replace ASQL_DB with the name of the Shared Database.

The Ready-to-Execute AltaSQL defined functions spreadsheet can be easily modified to use different databases by changing the Shared Database name. This enables parallel testing of different AltaSQL product offerings.

The spreadsheet can be downloaded from the AltaSQL.io website. See the Product pages.

Meta-data Repository Implementations

There are two types of meta-data repositories:

1. Meta-data Repository – a Snowflake TABLE whose data is treated primarily as meta-data for generating DDL, DML and documentation as well as a general-purpose meta-data catalog.
2. Virtual Repository – `a Snowflake VIEW referencing the Information Schema; it does not support transformation or AltaSQL Tag and requires no maintenance for schema changes. It only provides AltaSQL's replacement for SELECT *.`

Meta-Data Repository Categories[1]

Meta-data repositories fall into two broad categories: Administered Repositories and User-Managed Repositories. The categories and usage are customer defined; AltaSQL does not have pre-defined categories.

Physical and Virtual Meta-data Repositories

Meta-data repositories may be either:

- Physical tables
- Temporary tables
- VIEWs based on repository TABLEs.
    a. Care must be taken when creating VIEWs from multiple physical repositories to avoid View Name clashes

Administered Meta-data Repositories

Administered meta-data repositories' content and access are governed by an administrator. The contents are typically created by architects and developers, as well as submissions from power users and analysts. Administration should follow your organization's development standards and processes. Administrators should have a solid understanding of the meta-data repository structure and SQL.

AltaSQL provides many functions to manage the meta-data repository; in general, writing Snowflake SQL should not be required. However, administrators may need or choose to create Snowflake SQL for managing repositories.

Work product submitted for inclusion in Administered Repositories will typically be .CSV files. AltaSQL provides functions for extracting .CSV files with automatic protection of data with embedded commas. Staging tables should be used for loading .CSV files. The meta-data can then be directly inserted into the Administered Repository via INSERT or MERGE, or the .CSV can be loaded directly.

End users should have only SELECT access to managed repositories. They may also have CLONE privileges to create User-owned meta-data repositories.

End user Snowflake privileges determine how they can use a repository, e.g.:

1. Access to a schema with `CREATE TABLE` and possibly `CREATE VIEW` privileges. These privileges would typically be available to developers and power users.

[1] This section does not apply to Virtual Repositories

   a. This enables the user to `CREATE` repositories either for development and subsequent submission of generated SQL or new and revised repositories or View Definitions.
2. Access to a schema with SELECT privileges.. The user can generate SQL statements but cannot modify or create repositories or VIEW Definitions. The user can use generated statements, including using FROM_CLAUSE[2] and POST_FROM clauses where appropriate. Modified generated statements may be saved as a script or may also be saved as AltaSQL Tags.

User-Managed Meta-data Repositories

User-Managed Repositories are typically used by individuals or small teams, i.e., not under formal control. The user requires appropriate privileges on schemas (see above) for creating and managing repositories and appropriate privileges on other schemas for executing generated SQL statements. Such repositories may, of course, be shared on a more informal basis.

User-Managed repositories can also be used strictly for DML generation, primarily SELECT statements. This allows DML only users to effectively create reusable, tailored SELECT statements from existing Snowflake objects contained in a meta-data repository's View Definitions. The `SBLD_SELECT` functions enable the use of multiple `FROM_CLAUSEs` and `POST_FROM_CLAUSEs` by the user, which can be contained in AltaSQL Tags.

Security, PII and Anonymous Repositories

A core feature of AltaSQL repositories is comprehensive lineage of *generated DDL objects*, specifically VIEWs.

The comprehensive lineage repositories are those typically built for creating generally accessible Snowflake objects, e.g., governed/managed databases and schemas. Virtual Views do not use Dynamic Data Masks.

Although access is typically provided via Snowflake security policies, exposing the comprehensive lineage to end users may not be desirable for security reasons.

It is important to remember that ultimately, View Definitions for SQL statement generation refer to objects deployed in Snowflake.

It is a fairly simple process to reduce the comprehensive lineage repositories to simpler, "anonymous" repositories by removing information from repositories and ensuring that VIEWs referenced in the repository are SECURE VIEWs. Information to be removed may include:

- TABLEs as SOURCE_OBJECTs
- DATA_MASKS
- EXPRESSIONs

[2] FROM_CLAUSE is not supported in AltaSQL SELECT Discover

- UDF_CALLs
- View Definitions based on multiple SOURCE_OBJECTs

Tailored Usage Repositories

A key feature of serially reusable meta-data is the ability to create repositories with tailored usages. In addition to usage described in the previous section, repositories can be configured with different contents to limit, manage or provide convenience.

Working with Spreadsheets and .CSV Files

Working with AltaSQL meta-data in spreadsheets can be very convenient. Typical usage is to extract the desired meta-data as a .CSV file with the primary intent of creating new View Definitions with numerous custom edits. If the intent is updating existing repository meta-data, we recommend deleting and purging the View Definition rather than using a MERGE.

Once edited, the data is saved as a .CSV file to load into a repository.

This file can also be used for loading into Snowflake via COPY INTO from a staged file.

The function:

```
SBLD_FILE_FORMAT_CSV (
  FORMAT_NAME   VARCHAR,
  TO_DB  VARCHAR DEFAULT NULL,
  TO_SCHEMA     VARCHAR DEFAULT NULL,
  FILE_FORMAT_CONTROL VARCHAR DEFAULT NULL
  )
FILE_FORMAT_determines FILE type and CREATE option.
[
[TEMP | TEMPORARY | VOLATILE | TRANSIENT ]
[NOT_EXISTS |OR_REPLACE]
]
```

generates an appropriate CREATE FILE FORMAT statement.

IMPORTANT: While we have had no issues exporting .CSV files from Snowflake, we have encountered isolated instances where a SQL tool has decomposed a column value with multiple lines and quotes into multiple fields instead of a single field. This will result in load failure. Inspect the .CSV file to identify such issues.

Creating a Governed Meta-Data Repository Database

We recommend creating one or more Meta-data Repository Database (MDRDB). The script *CREATE_META_DATA_REPO_DB* is available for download from the AltaSQL.io website is a standard, reusable and easily modified script for creating a database to hold and govern both Administered and User managed repositories.

The script uses session variables to define names for all the various components. The script creates roles using the following session variables:

- Snowflake objects
    - $DB_NAME – name of the database
    - $SHARED_REPO_SCHEMA_NAME – repositories available to end users via $ANALYST_ROLE
    - $USER_REPO_SCHEMA_NAME – repositories for developers and power users (see …)
    - $DW_XSM – XSM warehouse for user of the database (see below)
- Account level ROLEs; may be assigned with SECURITYADMIN to USERs
    - $ADMIN _ROLE – owns the database and acts as both de facto SYSADMIN and SECURITYADMIN for GRANTing DATABASE ROLE to ACCOUNT LEVEL ROLEs.
    - $ANALYST_ROLE – SELECT on repositories in $SHARED_REPO_SCHEMA_NAME WITH GRANT OPTION
    - $DML_ROLE - SELECT,INSERT, UPDATE, DELETE, TRUNCATE on $SHARED_REPO_SCHEMA_NAME; $ADMIN_ROLE assumed to only role to CREATE TABLE/VIEW in $SHARED_REPO_SCHEMA
    - $CURRENT_USER_ROLE – role intended to be used as ownership role by current user
- DATABASE ROLEs assignable to ACCOUNT level ROLES by $ADMIN_ROLE
    - $DEV_ROLE – CREATE TABLE in $USER_REPO_SCHEMA and CREATE SCHEMA in $DB_NAME (see
    - $ANALYST_ROLE - SELECT on repositories in $SHARED_REPO_SCHEMA_NAME WITH GRANT OPTION
    - DML_ROLE - SELECT,INSERT, UPDATE, DELETE, TRUNCATE on $SHARED_REPO_SCHEMA_NAME; $ADMIN_ROLE assumed to be the only role to execute CREATE TABLE/VIEW in $SHARED_REPO_SCHEMA

This structure allows both the $ADMIN_ROLE and SECURITYADMIN ROLES TO GRANT the ACCOUNT level roles to USERs and other ACCOUNT LEVEL ROLEs. The latter can be used to add AltaSQL features to existing ROLEs with suitable access privileges to your existing data.

The $DEV_ROLE can only be GRANTed to USER by the $ADMIN_ROLE.

The $SHARED_REPO_SCHEMA provides a common source of ROW ACESS POLICIES. Note that the `FUNCTION SBLD_REPO_FROM_INFO_SCHEMA` has an optional argument `ROW_ACCESS`.

We recommend that repositories initially created from the Snowflake INFORMATION SCHEMA be created in the $SHARED_REPO_SCHEMA; this provides the basic SELECT functionality and reduces the need for developers and power users to create redundant repositories.

Self-Provisioning with User ROLES

The use of the DATABASE ROLE $DEV_ROLE is designed to enable self-provisioning of repositories and schemas within the governed database. In this script, $DEV_ROLE can only be assigned to an Account level role by the $ADMIN_ROLE. This assumes the existence or creation of a ROLE that is typically only assigned to a given user. The recommended usage is to create user managed repositories in the $USER_REPO_SCHEMA_NAME schema. The intent of allowing user owned SCHEMAs is for testing of generated DDL.

Note that you can change the Role Based Access Control (RBAC) structure and usage per your needs and standards.