Data Storage Considerations

This topic provides guidelines and best practices for controlling data storage costs associated with Continuous Data Protection (CDP), particularly for tables.

CDP, which includes Time Travel and Fail-safe, is a standard set of features available to all Snowflake accounts at no additional cost. However, because your account is charged for all data stored in tables, schemas, and databases created in the account, CDP does have an impact on storage costs, based on the total amount of data stored and the length of time the data is stored.

Storage is calculated and charged for data regardless of whether it is in the Active, Time Travel, or Fail-safe state. Because these life-cycle states are sequential, updated/deleted data protected by CDP will continue to incur storage costs until the data leaves the Fail-safe state.

Note
TIME_TRAVEL_BYTES and FAILSAFE_BYTES will incur charges when you load data using INSERT, COPY or SNOWPIPE. That's because small micro-partition defragmentation deletes small micro-partitions and creates a new micro-partition that has the same data. The deleted micro-partitions contribute to TIME_TRAVEL_BYTES and FAILSAFE_BYTES.

Monitoring Data Storage

Storage for Your Account (Account Administrators Only)

If you have been assigned the ACCOUNTADMIN role (i.e. you serve as the top-level administrator for your Snowflake account), you can use Snowsight or the Classic Console to view data storage across your entire account:

Snowsight
Select **Admin** » **Cost Management** » **Consumption**.

Classic Console

Click on **Account** » **Billing & Usage** » **Average Storage Used**

This page displays the total average data storage for your account, as well as the total for all databases, internal and named stages, and data in Fail-safe.

For more information, see Exploring storage cost.

Individual Table Storage

Any user with the appropriate privileges can view data storage for individual tables. Snowflake provides the following methods for viewing table data storage:

Classic Console

Click on **Databases** » *<db_name>* » **Tables**

SQL
Execute a SHOW TABLES command.

or

Query either of the following:

- TABLE_STORAGE_METRICS view (in the Snowflake Information Schema).
- TABLE_STORAGE_METRICS View view (in Account Usage).

Of the three methods, TABLE_STORAGE_METRICS provides the most detailed information because it includes a breakdown of the physical storage (in bytes) for table data in the following three states of the CDP life-cycle:

- Active (ACTIVE_BYTES column)
- Time Travel (TIME_TRAVEL_BYTES column)
- Fail-safe (FAILSAFE_BYTES column)

The view also provides columns for distinguishing between owned storage and referenced storage that occurs when cloning tables (see section below).

Staged File Storage (for Data Loading)

To support bulk loading of data into tables, Snowflake utilizes stages where the files containing the data to be loaded are stored. Snowflake supports both internal stages and external stages.

Data files staged in Snowflake internal stages are not subject to the additional costs associated with Time Travel and Fail-safe, but they do incur standard data storage costs. As such, to help manage your storage costs, Snowflake recommends that you monitor these files and remove them from the stages once the data has been loaded and the files are no longer needed. You can choose to remove these files either during data loading (using the COPY INTO <table> command) or afterwards (using the REMOVE command).

For more information, see Data loading considerations.

Tip
Periodic purging of staged files can have other benefits, such as improved data loading performance.

Cloning Tables, Schemas, and Databases

Snowflake's zero-copy cloning feature provides a convenient way to quickly take a "snapshot" of any table, schema, or database and create a derived copy of that object which initially shares the underlying storage. This can be extremely useful for creating instant backups that do not incur any additional costs (until changes are made to the cloned object).

However, cloning makes calculating total storage usage more complex because each clone has its own separate life-cycle. This means that changes can be made to the original object or the clone independently of each other and these changes are protected through CDP.

For example, when a clone is created of a table, the clone utilizes no data storage because it shares all the existing micro-partitions of the original table at the time it was cloned; however, rows can then be added, deleted, or updated in the clone independently from the original table. Each change to the clone results in new micro-partitions that are owned exclusively by the clone and are protected through CDP.

In addition, clones can be cloned, with no limitations on the number or iterations of clones that can be created (e.g. you can create a clone of a clone of a clone, and so on), which results in a n-level hierarchy of cloned objects, each with their own portion of shared and independent data storage.

Table IDs

Every Snowflake table has an ID that uniquely identifies the table. In addition, every table is also associated with a CLONE_GROUP_ID. If a table has no clones, then the ID and CLONE_GROUP_ID are identical. These IDs are displayed in the TABLE_STORAGE_METRICS view.

Owned Storage vs Referenced Storage

When a table is cloned, it is assigned a new ID and the CLONE_GROUP_ID for the original table. At the instant the clone is created, all micro-partitions in both tables are fully shared. The storage associated with these micro-partitions is ***owned*** by the oldest table in the clone group and the clone ***references*** these micro-partitions.

After a clone is created, both tables within the clone group have separate life-cycles, such that any DML operations on either table create new micro-partitions that are owned by their respective tables. Storage associated with these micro-partitions can be queried using the RETAINED_FOR_CLONE_BYTES column in the TABLE_STORAGE_METRICS view.

Because every table within a clone group has an independent life-cycle, ownership of the storage within these tables sometimes needs to be transferred to a different table within the clone group. For example, consider a clone group that consists of:

Original table:		Cloned to:		Cloned to:
T1	»	**T2**	»	**T3**

If T2 and T3 share some micro-partitions and T2 is dropped, then ownership of that storage must be transferred before T2 enters Fail-safe. In Snowflake, this transfer occurs at the time the micro-partitions exit the Time Travel state and would otherwise enter Fail-safe. In the case above, the micro-partitions that were previously owned by T2 are transferred to T3 when the Time Travel retention period expires.

Managing Costs for Short-Lived Tables

CDP is designed to provide long-term protection for your data. This data is typically stored in permanent tables. Unless otherwise specified at the time of their creation, tables in Snowflake are created as permanent.

During an ETL or data modeling process, tables may be created that are short-lived. For these tables, it does not make sense to incur the storage costs of CDP. Snowflake provides two separate mechanisms to support short-lived tables:

- Temporary tables
- Transient tables

Temporary Tables

Similar to other SQL databases, a temporary table exists only within a single user session and only within the duration of the session. Snowflake temporary tables have no Fail-safe and have a Time Travel retention period of only 0 or 1 day; however, the Time Travel period ends when the table is dropped.

Thus, the maximum total CDP charges incurred for a temporary table are 1 day (or less if the table is explicitly dropped or dropped as a result of terminating the session). During this period, Time Travel can be performed on the table.

Important
A connection and a session are different concepts within Snowflake. When logged into Snowflake, one or more sessions may be created. A Snowflake session is only terminated if the user explicitly terminates the session or the session times out due to inactivity after 4 hours. Disconnecting from Snowflake does not terminate the active sessions. Thus, a Snowflake session may be very long-lived and any temporary tables created within that session will continue to exist until they are dropped or the session is terminated.

To avoid unexpected storage costs for temporary tables, Snowflake recommends creating them as needed within a session and dropping them when they are no longer required.

Transient Tables

Transient tables are unique to Snowflake. They have characteristics of both permanent and temporary tables:

- In contrast to temporary tables, transient tables are not associated with a session; they are visible to all users who have permissions to access that table. Also, similar to permanent tables, they persist beyond the session in which they were created.
- In keeping with temporary tables, transient tables have no Fail-safe and have a Time Travel retention period of only 0 or 1 day.

Thus, the maximum total CDP charges incurred for a transient table are 1 day. During this period, Time Travel can be performed on the table.

Managing Costs for Large, High-Churn Tables

In data platforms, tables are typically either *fact* or *dimension* tables, which have different usage patterns and, therefore, different storage considerations:

- Fact tables are typically very large in size and experience a low degree of churn (row updates or deletes). Most changes to fact tables are inserts of new data or, in some cases, deletions of older data. CDP is ideal for fact tables as it provides full data protection at a very low storage cost.
- Dimension tables have a different update pattern. Row updates and deletions are much more common in dimension tables. When one or more rows of a table are updated or deleted, the underlying micro-partitions that store this data begin the life-cycle transitions associated with CDP. For high-churn dimension tables, the resulting storage associated with Time Travel and Fail-safe data can be much larger than the active table storage.

For the vast majority of dimension tables, the CDP storage cost associated with these updates are reasonable. Dimension tables are usually small in size and even if frequently updated, the cost of storage in Snowflake is inexpensive and the benefits of CDP far outweigh the costs.

For some larger, high-churn dimension tables, the storage costs associated with CDP can be significant. When multiple updates are made to a table, all of the impacted micro-partitions are re-created and then they transition through the CDP storage life-cycle.

High-churn dimension tables can be identified by calculating the ratio of FAILSAFE_BYTES divided by ACTIVE_BYTES in the TABLE_STORAGE_METRICS view. Any table with a large ratio is considered to be a high-churn table. Because storage in Snowflake is inexpensive and most high-churn tables consume a modest amount of total storage, even if the ratio is high, the preferred option is to create these tables as permanent and use CDP to protect the data.

In some cases, the cost of storage for high-churn dimension tables is excessive and you might prefer an alternative option to CDP. As an extreme example, consider a table with rows associated with every micro-partition within the table (consisting of 200 GB of physical storage). If every row is updated 20 times a day, the table would consume the following storage:

Active
200 GB

Time Travel
4 TB

Fail-safe
28 TB

Total Storage
32.2 TB

For large, high-churn dimension tables that incur overly-excessive CDP costs, the solution is to create these tables as transient with zero Time Travel retention (i.e. DATA_RETENTION_TIME_IN_DAYS=0) and then copy these tables on a periodic basis into a permanent table. This effectively creates a full backup of these tables. Because each backup is protected by CDP, when a new backup is created, the old one can be deleted.

Using the example above, the storage costs associated with the same 200 GB, high-churn dimension table that was backed up once a day would be:

Active
200 GB

Time Travel
200 GB

Fail-safe
1.4 TB

Backup
200 GB

Total Storage
2 TB

Tip
The backups should be performed as often as necessary to ensure full recovery in the event of data loss. For these tables, Snowflake recommends backups be taken at least once a day.