About Secure Data Sharing

Secure Data Sharing lets you share selected objects in a database in your account with other Snowflake accounts. You can share the following Snowflake objects:

- Databases
- Tables
- Dynamic tables
- External tables
- Apache Iceberg™ tables
- Secure views
- Secure materialized views
- Secure user-defined functions (UDFs)
- Models without code (such as CORTEX_FINETUNED) - *Preview*

Snowflake enables the sharing of databases through *shares*, which are created by data providers and “imported” by data consumers.

Important
All database objects shared between accounts are ***read-only*** (i.e. the objects cannot be modified or deleted, including adding or modifying table data).

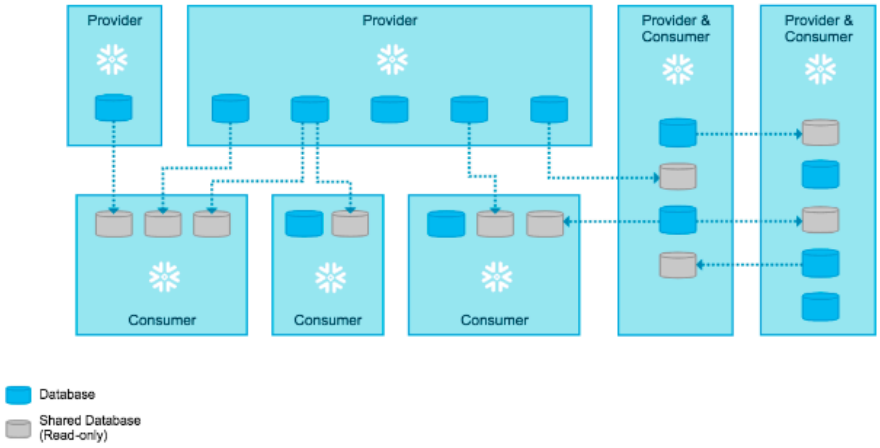
How does Secure Data Sharing work?

With Secure Data Sharing, ***no*** actual data is copied or transferred between accounts. All sharing uses Snowflake’s services layer and metadata store. Shared data does not take up any storage in a consumer account and therefore does not contribute to the consumer’s monthly data storage charges. The ***only*** charges to consumers are for the compute resources (i.e. virtual warehouses) used to query the imported data.

Because no data is copied or exchanged, Secure Data Sharing setup is quick and easy for providers and access to the imported data is near-instantaneous for consumers:

- The provider creates a share of a database in their account and grants access to specific objects in the database. The provider can also share data from multiple databases, as long as these databases belong to the same account. One or more accounts are then added to the share, which can include your own accounts (if you have multiple Snowflake accounts).
  For more details, refer to What is a share? (in this topic).
- On the consumer side, a ***read-only*** database is created from the share. Access to this database is configurable using the same, standard role-based access control that Snowflake provides for all objects in the system.

With this architecture, Snowflake enables a network of providers that can share data with multiple consumers (including within their own organization) and consumers that can access imported data from multiple providers:



Note
Any full Snowflake account can both provide and consume imported data. Snowflake also supports third-party accounts, a special type of account that consumes imported data from a single provider account. For more details, refer to Reader accounts for third-party access (in this topic).

What is a share?

Shares are named Snowflake objects that encapsulate all of the information required to share a database.

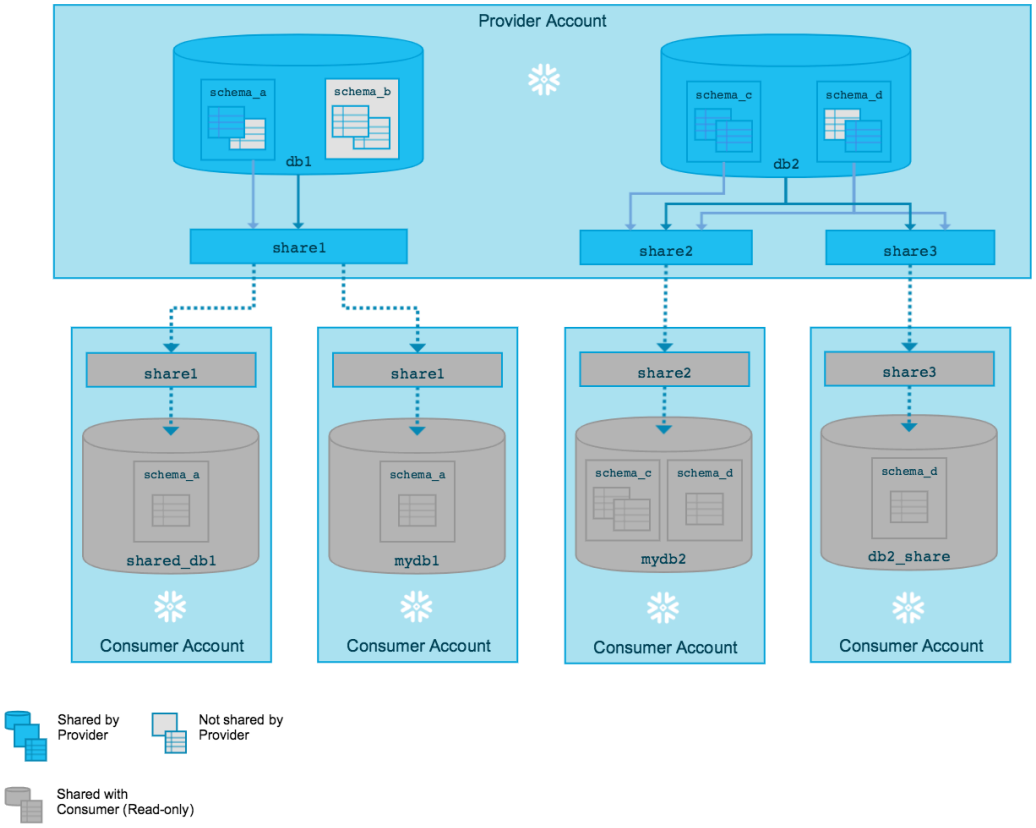
Data providers add Snowflake objects (databases, schemas, tables, secure views, etc.) to a share using ***either or both*** of the following options:

- **Option 1:** Grant privileges on objects to a share via a database role.
- **Option 2:** Grant privileges on objects directly to a share.

For more information on these options, refer to How to share database objects.

You choose which accounts can consume data from the share by adding the accounts to the share.

After a database is created (in a consumer account) from a share, all the imported objects are accessible to users in the consumer account:



Shares are secure, configurable, and controlled completely by the provider account:

- New objects added to a share become immediately available to all consumers, providing real-time access to imported data.
- Updates to existing objects in a share become immediately available to all consumers.
- Access to a share (or any of the objects in a share) can be revoked at any time.

Options for sharing in Snowflake

You can share data in Snowflake using one of the following options:

- a Listing, in which you offer a share and additional metadata as a data product to one or more accounts,
- a Direct Share, in which you directly share specific database objects (a share) to another account in your region,
- a Data Exchange, in which you set up and manage a group of accounts and offer a share to that group.

Using SQL with data shares

Preparing objects to share can be performed using any role. Other data sharing tasks, such as creating a share or adding consumer accounts to the share, requires the ACCOUNTADMIN role or a role granted the global CREATE SHARE privilege. For more details about the CREATE SHARE privilege, see Enable non-ACCOUNTADMIN roles to perform data sharing tasks.

If you want to use DDL to create and manage database roles, use the commands listed here:

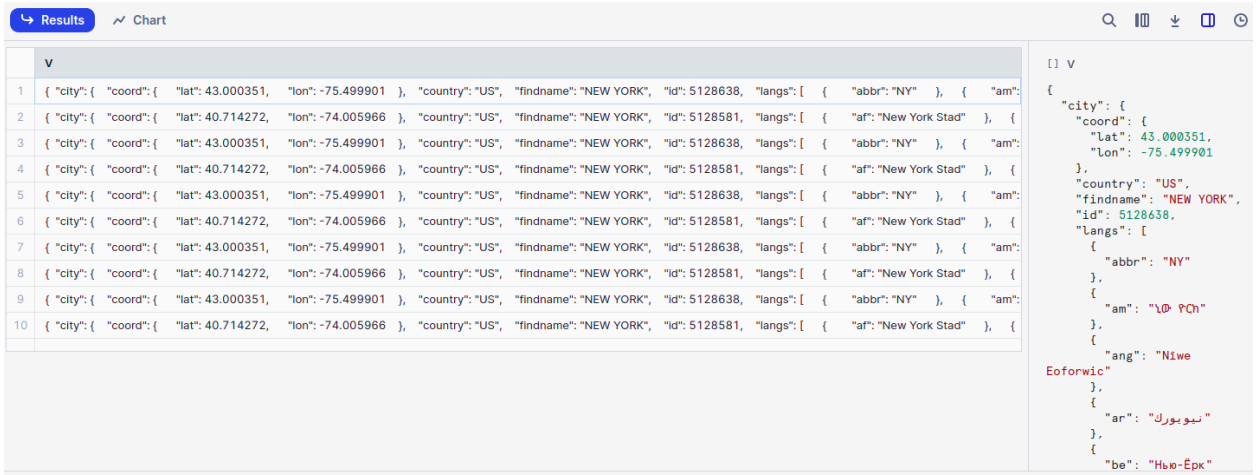
- CREATE DATABASE ROLE
- ALTER DATABASE ROLE
- DROP DATABASE ROLE
- SHOW DATABASE ROLES
- A shared database role does not support future grants on objects. For details, see GRANT DATABASE ROLE … TO SHARE.

If you want to use DDL to view, grant, or revoke access to database objects in a share, use the commands listed here:

- GRANT DATABASE ROLE … TO SHARE
- REVOKE DATABASE ROLE … FROM SHARE
- GRANT <privilege> … TO SHARE
- REVOKE <privilege> … FROM SHARE
- SHOW GRANTS TO SHARE … — lists all object privileges that have been granted to a share
- SHOW GRANTS OF SHARE … — lists all accounts for the share and indicates the accounts that are using the share

To begin, let's create a new database and some tables. We are going to create a database with weather data:

```
--Lab 9.1
--Creating a weather database

use role accountadmin;
create or replace database weather;

use warehouse compute_wh;
use database weather;
use schema public;

create table json_weather_data (v variant);

create stage nyc_weather
url = 's3://snowflake-workshop-lab/weather-nyc';

list @nyc_weather;

copy into json_weather_data
from @nyc_weather
file_format = (type=json);
select * from json_weather_data limit 10;
```



Results | Chart

	v
1	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORK", "id": 5128638, "langs": [{ "abbr": "NY" }, { "am":
2	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORK", "id": 5128581, "langs": [{ "af": "New York Stad" }, {
3	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORK", "id": 5128638, "langs": [{ "abbr": "NY" }, { "am":
4	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORK", "id": 5128581, "langs": [{ "af": "New York Stad" }, {
5	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORK", "id": 5128638, "langs": [{ "abbr": "NY" }, { "am":
6	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORK", "id": 5128581, "langs": [{ "af": "New York Stad" }, {
7	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORK", "id": 5128638, "langs": [{ "abbr": "NY" }, { "am":
8	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORK", "id": 5128581, "langs": [{ "af": "New York Stad" }, {
9	{ "city": { "coord": { "lat": 43.000351, "lon": -75.499901 }, "country": "US", "findname": "NEW YORK", "id": 5128638, "langs": [{ "abbr": "NY" }, { "am":
10	{ "city": { "coord": { "lat": 40.714272, "lon": -74.005966 }, "country": "US", "findname": "NEW YORK", "id": 5128581, "langs": [{ "af": "New York Stad" }, {

[] v


```
{
  "city": {
    "coord": {
      "lat": 43.000351,
      "lon": -75.499901
    },
    "country": "US",
    "findname": "NEW YORK",
    "id": 5128638,
    "langs": [
      {
        "abbr": "NY"
      },
      {
        "am": "ኒውዮርክ"
      },
      {
        "ang": "Niwe
Eoforwic"
      },
      {
        "ar": "نيويورك"
      },
      {
        "be": "Нью-Ёрк"
```

```sql
-- Create table

create or replace table weather_data as
select
  v:time::timestamp as observation_time,
  v:city.id::int as city_id,
  v:city.name::string as city_name,
  v:city.country::string as country,
  v:city.coord.lat::float as city_lat,
  v:city.coord.lon::float as city_lon,
  v:clouds.all::int as clouds,
  (v:main.temp::float)-273.15 as temp_avg,
  (v:main.temp_min::float)-273.15 as temp_min,
  (v:main.temp_max::float)-273.15 as temp_max,
  v:weather[0].main::string as weather,
  v:weather[0].description::string as weather_desc,
  v:weather[0].icon::string as weather_icon,
  v:wind.deg::float as wind_dir,
  v:wind.speed::float as wind_speed
from json_weather_data;

-- Check the data
select * from weather_data limit 10;


--Let's create a view of the unique weather patterns
create or replace view weather_type as
select distinct weather
from weather_data;

-- Check the data
select * from weather_type;
```

Results | Chart

	OBSERVATION_TIME	CITY_ID	CITY_NAME	COUNTRY	CITY_LAT	CITY_LON	CLOUDS	TEMP_AVG
1	2016-10-26 10:31:26.000	5128638	New York	US	43.000351	-75.499901	80	0.726
2	2016-10-26 10:34:03.000	5128581	New York	US	40.714272	-74.005966	0	4.101
3	2016-11-06 11:37:03.000	5128638	New York	US	43.000351	-75.499901	90	6

Results | Chart

	WEATHER
1	Clouds
2	Clear
3	Mist
4	Rain
5	Haze
6	Drizzle
7	Thunderstorm
8	Snow
9	Smoke
10	Dust
11	Squall
12	Fog

```
--Let's create a view of the highest temperatures by city
create or replace view weather_max as
select city_name
, max(temp_max) as temperature
from weather_data
group by city_name;

-- Check the data
select * from weather_max;
```

	CITY_NAME	TEMPERATURE
1	New York	38

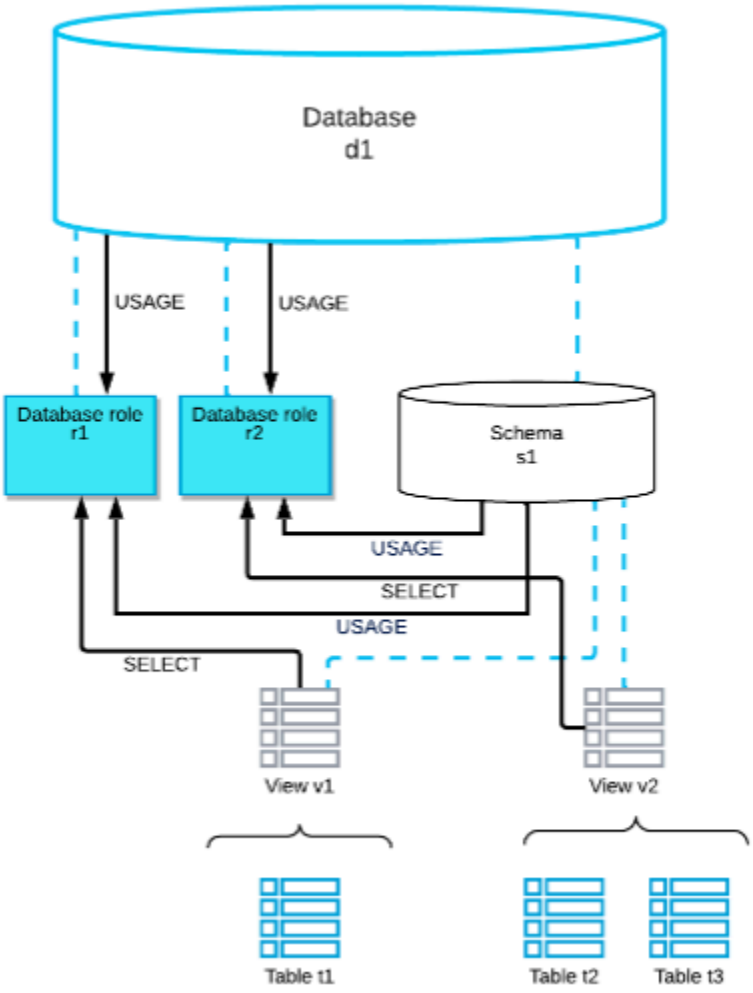
Grant database roles to a share

This section provides instructions for data providers to restrict access to databases and database objects in a share using database roles.

Note
To perform the tasks described in this topic, your role must have the global CREATE DATABASE and CREATE SHARE privileges.
In the extended example throughout this section, a data provider shares the following objects with data consumers:

The data provider creates two database roles in database `weather` to control access to these objects: `weather.r1` and `weather.r2`.

The following diagram shows the relationships among these objects and indicates the privileges that are be granted to the database roles:



Create database roles

Create a new database role or replace an existing database role using CREATE DATABASE ROLE.

For example, create database roles `weather.r1` and `weather.r2` using fully-qualified identifiers:

```
--Lab 9.2

CREATE DATABASE ROLE weather.r1;
CREATE DATABASE ROLE weather.r2;
```

Grant privileges on objects to database roles

Grant privileges on a single database and subset of objects in the database to each database role using GRANT <privileges>. Only grant privileges on objects that the database role should allow access to.

Either specify the fully-qualified name of a database role, or set the database as the active database in a session and then specify the relative name.

Note

- To perform the tasks described in this topic, you must use the ACCOUNTADMIN role or a role granted the relevant privileges. For more information, including additional data sharing scenarios, see Create and configure shares.
- Privileges granted to a database role are limited to USAGE on the database and schema that contain the database role and privileges on other objects in the same database. In particular, note that the REFERENCE_USAGE privilege cannot be granted to a database role to include objects from multiple databases in a share.

The following SQL statements grant the privileges to the `weather.r1` database role:

```
GRANT USAGE ON SCHEMA weather.public TO DATABASE ROLE weather.r1;
GRANT SELECT ON VIEW weather.public.weather_type TO DATABASE ROLE weather.r1;
```

The following SQL statements grant the privileges to the `weather.r2` database role:

```
GRANT USAGE ON SCHEMA weather.public TO DATABASE ROLE weather.r2;
GRANT SELECT ON VIEW weather.public.weather_max TO DATABASE ROLE weather.r2;
```

Granting the USAGE privilege on the parent database is not necessary. This privilege is granted implicitly when a database role is created.

To view all privileges granted to a database role, execute SHOW GRANTS TO DATABASE ROLE using fully-qualified identifiers:

```
SHOW GRANTS TO DATABASE ROLE weather.r1;
SHOW GRANTS TO DATABASE ROLE weather.r2;
```

	created_on	privilege	granted_on	name	granted_to	grantee_name	grant_option	granted_by
1	2024-11-02 07:10:28.910 -0700	USAGE	DATABASE	WEATHER	DATABASE_ROLE	R1	false	
2	2024-11-02 07:10:29.674 -0700	USAGE	SCHEMA	WEATHER.PUBLIC	DATABASE_ROLE	R1	false	ACCOUNTADMIN
3	2024-11-02 07:10:29.951 -0700	SELECT	VIEW	WEATHER.PUBLIC.WEATHER_TYPE	DATABASE_ROLE	R1	false	ACCOUNTADMIN

Create a share

Create a share using CREATE SHARE. The share is an empty container at this stage in the process.

For example, create a new share named `share1`:

```
CREATE SHARE share1;
```

Add the database by granting the USAGE privilege to the share

Currently, it is necessary to grant the USAGE privilege on a database to include it in a share.

For example, grant the USAGE privilege on the `weather` database to share `share1`:

```
GRANT USAGE ON DATABASE weather TO SHARE share1;
```

Add objects by granting database roles to the share

Add databases and database objects to a share by granting database roles to the share using GRANT DATABASE ROLE … TO SHARE.

For example, grant database roles `weather.r1` and weather`1.r2` to share `share1`:

```
GRANT DATABASE ROLE weather.r1 TO SHARE share1;
GRANT DATABASE ROLE weather.r2 TO SHARE share1;
```

Share the database objects with one or more data consumer accounts

Modify the share ALTER SHARE … ADD ACCOUNTS and add database consumer accounts with which you want to share the database objects.

The following example adds accounts `consumer1` and `consumer2` in organization `org1` to share `share1`:

```
ALTER SHARE share1 ADD ACCOUNTS = <org1.consumer1>,<org1.consumer2>;
```