Snowflake_Stream & Task Hands On | Complete Beginners Tutorial | Step By Step Guide

Hands-on tutorial to ingest CDC & Delta data using snowflake Stream & Task. This guide also helps you to understand how to ingest data continuously and transform it.

Snowflake Stream And Task Beginner's Guide

What is Stream (Object) in Snowflake?

A stream object records/captures data manipulation language (DML) changes made to a table, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data (rows & fields). This process is referred to as change data capture (CDC). An individual table stream tracks the changes made to rows in a source table. A table stream (also referred to as simply a "stream") makes a "change table" available of what changed, at the row level, between two transactional points of time in a table. Stream also allows querying and consuming a sequence of change records in a transactional fashion.

What is a Task Object in Snowflake?

A Snowflake Task (also referred to as simply a Task) is such an object that can schedule an SQL statement to be automatically executed as a recurring event.A task can execute a single SQL statement, including a call to a stored procedure. There is no event source that can trigger a task; instead, a task runs on a schedule. A task can have multiple tasks as its offsprings (task tree) (but not DAG. So, you can consider that it can do "fork", but there is no "join".

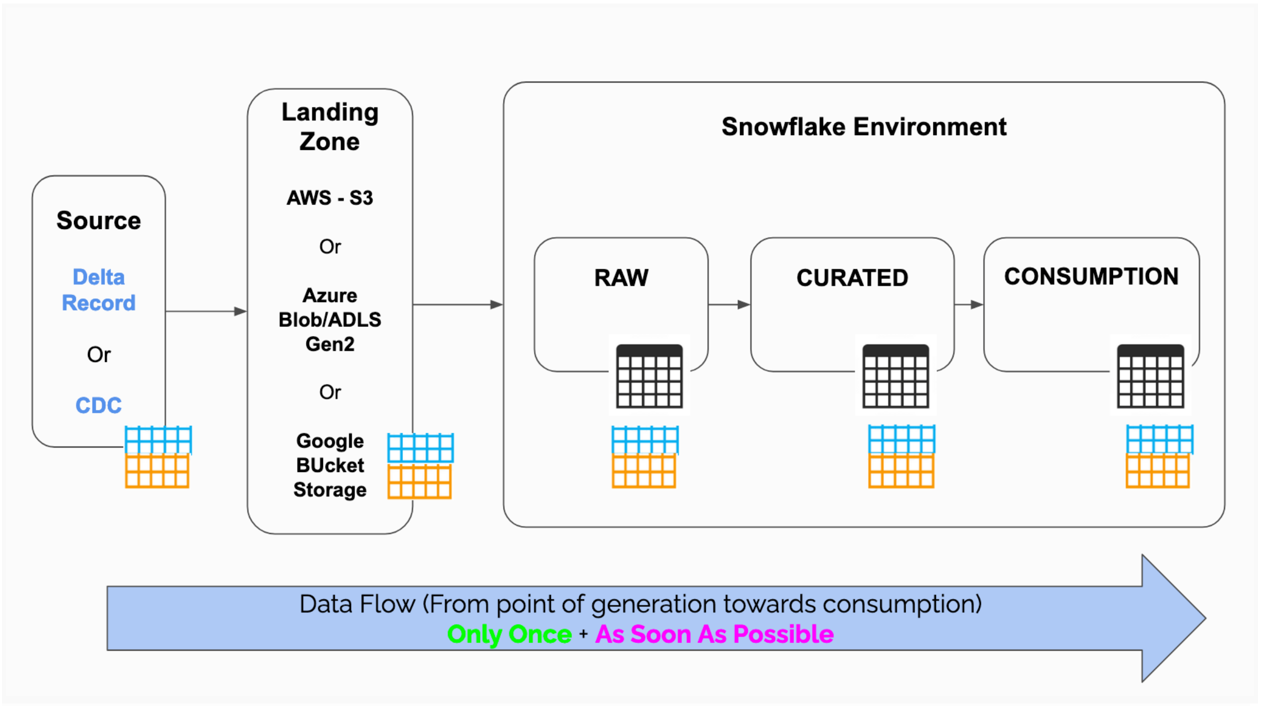
Snowflake Stream & Task Video Tutorial

Data Flow (From point of generation towards consumption)

The typical data ingestion flow

- **Source like RDBMS or any other event generator system which either relay the delta as and when it is produced or in some cases they are pushed in batches to improve the performance and reduce the transport cost.**

- Landing zone is the area where the event or cdc or delta data arrives first. In case of snowflake, generally this area is either a cloud storage (S3 or Azure or Google Bucket)
- Snowflake (Raw/curated/consumption) - This layer is the 1st entry point for snowflake.

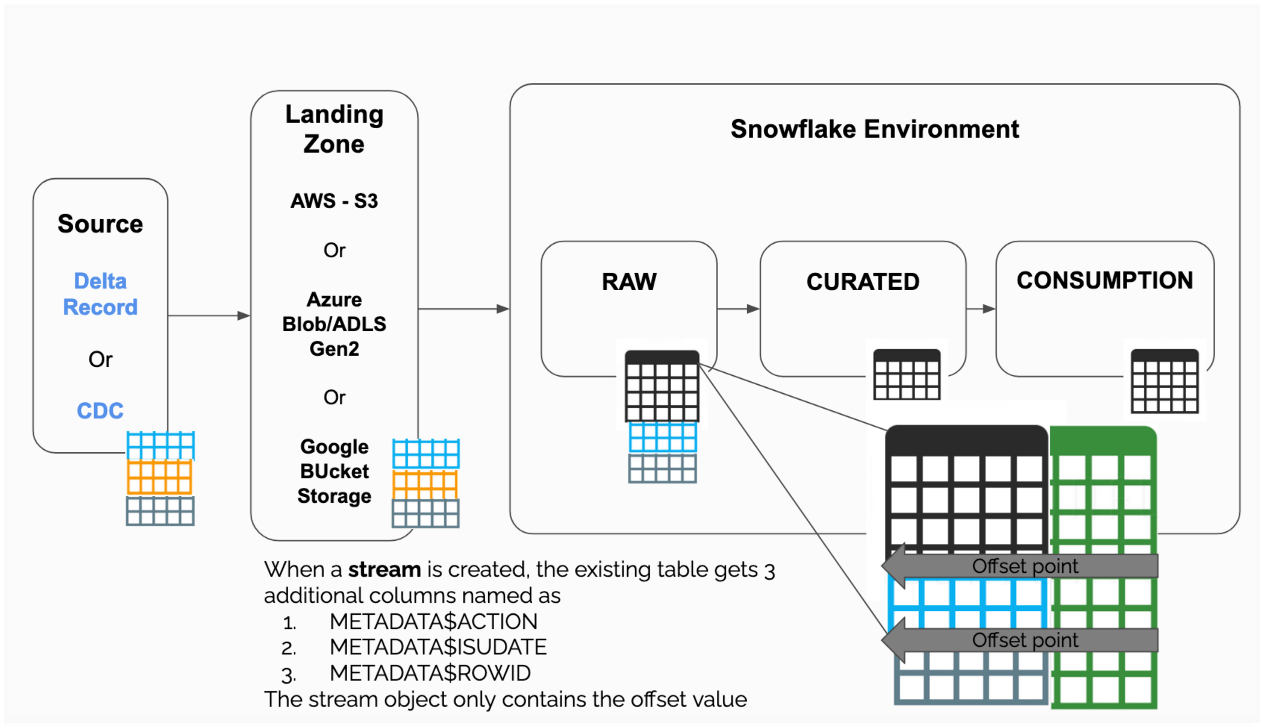


Metadata Columns for stream object

When a stream is created, the stream object just hold the offset from where the delta data or CDC can be tracked, however the main data setup is still with the primary table where object is created.

There are 3 additional columns which are added to the primary table where stream object is created

1. METATADADATA$ACTION: may have only two values Inert/Delete
2. METATADADATA$ISUPDATE: This will flagged as True if the record is update
3. METATADADATA$ROW_ID: This is the unique hashkeye which will be tracked against each changes.



When a **stream** is created, the existing table gets 3 additional columns named as

1. METADATA$ACTION
2. METADATA$ISUDATE
3. METADATA$ROWID

The stream object only contains the offset value

Sample Sales & Product Master Data for Stream & Task

Following sample data can be used to test your stream and task objects. There are just handful of sample data set and if needed, you must be able to extende it as per your need.

Alternatively, you can also download the data-subset from this link

```
-- Sales Transaction Table
insert into demo_db.public.sales_raw values (101,'2020-01-01',10001,1,111.11 );
insert into demo_db.public.sales_raw values (102,'2020-01-02',10002,2,222.22 );
insert into demo_db.public.sales_raw values (103,'2020-01-03',10003,3,333.33 );
insert into demo_db.public.sales_raw values (104,'2020-01-04',10004,4,444.44 );

insert into demo_db.public.sales_raw values (155,'2020-01-25',10005,0,555.55 ); -- will be modified later
insert into demo_db.public.sales_raw values (166,'2020-01-26',10006,0,600.06 ); -- will be modified later
insert into demo_db.public.sales_raw values (177,'2020-01-27',10007,0,777.77 ); -- will be modidfied later

insert into demo_db.public.sales_raw values (200,'2020-01-28',10008,3,0.11 );   -- will be deleted later
insert into demo_db.public.sales_raw values (200,'2020-01-29',10009,3,0.11 );   -- will be
deleted later
insert into demo_db.public.sales_raw values (200,'2020-01-30',10010,3,0.11 );   -- will be
deleted later

-- Product Master Data
insert into demo_db.public.sales_raw values (101,'Abbas MA-01','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (102,'Fama UE-85','Urban','All
Extreme','1','Fama');
insert into demo_db.public.sales_raw values (103,'Abbas MA-03','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (104,'Abbas MA-04','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (105,'Abbas MA-05','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (106,'Abbas MA-06','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (107,'Abbas MA-07','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (108,'Abbas MA-08','Mix','All
Season','1','Abbas');
insert into demo_db.public.sales_raw values (109,'Abbas MA-09','Mix','All
Season','1','Abbas');
```

Table DDL Statements

```
use role sysadmin;
use warehouse compute_wh;
use schema demo_db.public;

create or replace table demo_db.public.sales_ext(
  product_id string,
  purchase_date string,
  zip string,
  units string,
  revenue string
);

create or replace table demo_db.public.product_master(
  product_id int,
  product_desc varchar(),
  category varchar(10),
  segment varchar(20),
  manufacture_id int,
  manufacture varchar(50)
);

create or replace table demo_db.public.sales_raw(
  product_id int,
  purchase_date date,
  zip varchar(),
  units int,
  revenue decimal(10,2)
);

create or replace sequence demo_db.public.sales_sequence start = 1 increment = 1;
create or replace table demo_db.public.sales_consumption(
  tx_key number  default demo_db.public.sales_sequence.nextval,
  product_id int ,
  product_desc varchar(),
  category varchar(10),
  segment varchar(20),
  manufacture varchar(50),
  purchase_date date,
  zip varchar(),
  units int,
  revenue decimal(10,2)
);
```

Stage Objects & Insert Operation SQL Scripts

Following SQL Script creates stage objects & insert operation as select.

```
-- Stage Object
create or replace stage demo_db.public.sales_ext_stage
comment = 'This is my internal stage for loading data to sales external table (TopperTips)';

-- insert as select sql
insert into demo_db.public.sales_consumption
(product_id,product_desc,category,segment,manufacture,purchase_date, zip, units,
revenue)
select
  s.product_id,
  pm.product_desc,
  pm.category,
  pm.segment,
  pm.manufacture,
  s.purchase_date,
  s.zip,
  s.units,
  s.revenue
from demo_db.public.sales_raw s join demo_db.public.product_master pm
on s.product_id  = pm.product_id;
```

Stream & Task Objects

Following stream & task SQL script will help you to construct stream & task objects.

```
-- insert only stream
 create or replace stream demo_db.public.sales_raw_stream
 on table demo_db.public.sales_raw
 append_only=true
 comment = 'Insert only stream on sales raw  table';

  insert into demo_db.public.sales_raw
      select product_id,purchase_date,zip,units,revenue
      from demo_db.public.sales_ext
      where purchase_date in  ('2003-06-01','2003-05-31');

create or replace task demo_db.public.sales_task
   warehouse = compute_wh
   schedule  = '1 minute'
 when
   system$stream_has_data('demo_db.public.sales_raw_stream')
 as
   insert into demo_db.public.sales_consumption
   (product_id,product_desc,category,segment,manufacture,purchase_date, zip, units,
revenue)
   select
     s.product_id,
     pm.product_desc,
     pm.category,
     pm.segment,
     pm.manufacture,
     s.purchase_date,
     s.zip, s.units, s.revenue
   from demo_db.public.sales_raw_stream s join demo_db.public.product_master pm
   on s.product_id  = pm.product_id;
```