# Change Data Capture in Snowflake

## Introduction

Snowflake provides **data warehouse-as-a-service**. Its architecture is purely Cloud Native. It can scale up/down according to users and queries in a few seconds. Snowflake is changing expectations for **speed** and **flexibility** of a data warehouse. Nowadays Snowflake is one the prominent Data-warehousing solutions with no upfront cost. There are various benefits or features of Snowflake which makes it better than others like Zero Copy Cloning, Time Travel etc. Our focus in this article is to discuss one of its features through which it can implement Change Data Capture Functionality. As data in your system would change frequently, if you have run any script then in that scenario it would be a byzantine task to look after the entire process. Streams and Tasks in Snowflake make **Change Data Capture(CDC)** easy and effective by just using a few commands.

### Learning Objectives

- What is Change Data Capture(CDC)?
- What are Snowflake Streams?
- Why use CDC?
- How to use CDC(Change Data Capture)?
- What is Change Data Capture(CDC)?

## What is Change Data Capture(CDC)?

A stream object records **data manipulation language (DML)** changes made to tables, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data. 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.

## What are Snowflake Streams?

A Stream adds changed data capture to Snowflake so you're gonna have your source table and imagine the source table will typically live in your staging environment and when you put the stream on it to enable change data capture every time you insert data insert, update or delete data in your source table the stream just captures the changes so imagine you have you know many hundreds of millions of rows or terabytes of data in a large table and then you're doing your daily loads you don't have to reload your entire target table so by capturing just the change data capture we can use the merge statement to merge just the changes from source to target.

## Why to use CDC?

1. Can load real-time data from transactional databases easily.
2. It doesn't affect/harm source data/system.
3. It changes expectations for speed and flexibility of a data warehouse.
4. Old batch ETL uploads achieve the objective of moving the data to a target that has high-latency approaches that cannot support the continuous data pipelines and real-time operational decision-making.

## How to use CDC(Capture Data Change)?

First of all, either login to Snowflake WebBased UI or SnowSQL. Then follow below steps:

### Create database CDC_Stream

```
create or replace database CDC_STREAM;
```

```
Use CDC_STREAM;
```

### Create a table to be the source

```
create or replace table members_source (
```

```
id int,
```

```
first_name varchar(255),
last_name varchar(255) )
```

Results Data Preview

Query ID SQL 117ms 1 rows

Filter result... Copy

| Row | status |
|---|---|
| 1 | Table MEMBERS_SOURCE successfully created. |

## Create a table to be the destination

```
create or replace table members_destination ( id int, first_name varchar(255), last_name varchar(255) );
```

## Create a stream to track changes to date in the MEMBERS table

```
create or replace stream member_stream on table members_source;
```

Anytime we make changes to source table which is the members source it's going to be populated in the stream until some data management command comes and consumes it

## Enter a couple records into source table

## Add some record

```
insert into members_source values (1,'Wayne','Bell');
insert into members_source values (2,'Anthony','Allen');
insert into members_source values (3,'Eric','Henderson');
insert into members_source values (4,'Jimmy','Smith');
insert into members_source values (5,'Diana','Wheeler');
insert into members_source values (6,'Karen','Hall');
insert into members_source values (7,'Philip','Rodriguez');
insert into members_source values (8,'Ashley','Bryant');
insert into members_source values (9,'Norma','Grant');
```

```
insert into members_source values (10,'Helen','Lewis');
insert into members_source values (11,'Larry','Mccoy');
insert into members_source values (12,'Emily','Wood');
insert into members_source values (13,'Patrick','Alvarez');
```

## Here's our source data

```
select * from members_source;
```

Results | Data Preview | Open History

Query ID | SQL | 123ms | 13 rows

Filter result... | Copy | Columns

| Row | ID | FIRST_NAME | LAST_NAME |
|---|---|---|---|
| 1 | 1 | Wayne | Bell |
| 2 | 2 | Anthony | Allen |
| 3 | 3 | Eric | Henderson |
| 4 | 4 | Jimmy | Smith |

## As you see destination it should be empty

```
select * from members_destination;
```

## View the change log in the stream

```
select * from member_stream;
```

Results | Data Preview | Open History

Query ID | SQL | 138ms | 13 rows

Filter result... | Copy | Columns

| Row | ID | FIRST_NAME | LAST_NAME | METADATA$ACTION | METADATA$ISUPDATE | METADATA$ROW_ID |
|---|---|---|---|---|---|---|
| 1 | 2 | Anthony | Allen | INSERT | FALSE | 43c22aeb4eea86e301bd519264c9015bae0b91d6 |
| 2 | 12 | Emily | Wood | INSERT | FALSE | 24c15cf4a10bc858d3fe2ad4408821cf6a58dd53 |
| 3 | 3 | Eric | Henderson | INSERT | FALSE | 78852fe96f3e4bc1b4cb89da45393c8b9d8a3772 |
| 4 | 6 | Karen | Hall | INSERT | FALSE | 607db31594aae07f543b7765e4c0451bd4215f1f |
| 5 | 5 | Diana | Wheeler | INSERT | FALSE | eee3f1714ee60b6ef0f31781e3e4d368b3d46c9a |
| 6 | 10 | Helen | Lewis | INSERT | FALSE | 2323e6d23daa40b9c89fbdb9a01d2378cfd5daba |

Here are the records and we had a couple of metadata fields some data action will say insert or delete, in this case we have inserts field only.

So, right now there are no records in our destination but they're sitting here in the stream and we're going to move it over with the merge statement

we'll have the merge State, we're gonna merge into destination everything that came from the source table but from the stream.

```
Use CDC_STREAM;

MERGE into members_destination as T

using (select *

from member_stream) AS S

ON T.id = s.id

when matched AND S.metadata$action = 'INSERT' AND
S.metadata$isupdate

THEN

update set T.first_name = S.first_name, T.last_name =
S.last_name

When matched

And S.metadata$action = 'DELETE' THEN DELETE

when not matched

And S.metadata$action = 'INSERT' THEN

INSERT (id,

first_name,

last_name)

VALUES (S.id,

S.first_name,

S.last_name);
```

Results Data Preview Open History

Query ID SQL 600ms 1 rows

Filter result... Copy Columns

| Row | number of rows inserted | number of rows updated | number of rows deleted |
|---|---|---|---|
| 1 | 13 | 0 | 0 |

let's execute the destination table and now the data came by just like we wanted it to and then if we go and look at our stream there's not going to be anything in it because the merge consumed the stream so the change they capture the stream will continue to record all the change that happens in your source table.

## View the results

```
select * from members_destination;
```

Results Data Preview

Query ID SQL 128ms 13 rows

Filter result... Copy

| Row | ID | FIRST_NAME | LAST_NAME |
|---|---|---|---|
| 1 | 4 | Jimmy | Smith |
| 2 | 9 | Norma | Grant |
| 3 | 2 | Anthony | Allen |
| 4 | 3 | Eric | Henderson |
| 5 | 1 | Wayne | Bell |
| 6 | 13 | Patrick | Alvarez |

## We're gonna update the first record from Bell we're gonna say Wright or ID equals to 1

```
update members_source

set last_name='Wright'

where id=1;
```

Results Data Preview Open History

Query ID SQL 263ms 1 rows

Filter result... Copy Columns

| Row | number of rows updated | number of multi-joined rows updated |
|---|---|---|
| 1 | 1 | 0 |

When we look at the stream it has two records insert and delete and you can see the insert has the Wright value that's the new value we want the other column metadata is update

Results Data Preview

Query ID SQL 213ms 2 rows

Filter result... Copy

| Row | ID | FIRST_NA... | LAST_NAME | METADATA$ACTION | METADATA$ISUPDATE | METADATA$ROW_ID |
|---|---|---|---|---|---|---|
| 1 | 1 | Wayne | Wright | INSERT | TRUE | a13ac58d6198d8f486ac47dd2ea982c3fb425bc8 |
| 2 | 1 | Wayne | Bell | DELETE | TRUE | a13ac58d6198d8f486ac47dd2ea982c3fb425bc8 |

## Let's execute our merge statement as shown below

```
Use CDC_STREAM;

MERGE into members_destination as T

using (select *

from member_stream

Where Not (metadata$action = 'DELETE' AND metadata$isupdate =
TRUE)) AS S

ON T.id = s.id

when matched AND S.metadata$action = 'INSERT' AND
S.metadata$isupdate

THEN

update set T.first_name = S.first_name, T.last_name =
S.last_name

When matched

And S.metadata$action = 'DELETE' THEN DELETE

when not matched

And S.metadata$action = 'INSERT' THEN

INSERT (id,

first_name,

last_name)

VALUES (S.id,

S.first_name,

S.last_name);
```

## Run merge command and let's take a look at destination table

## View the results

```
select * from members_destination;
```

Let delete the some data from the source table i.e we are going to delete the **Anthony**

```
delete from members_source where id = 2;
```

Results Data Preview

Query ID SQL 221ms 1 rows

Filter result... Copy

| Row | number of rows deleted.. |
|---|---|
| 1 | 1 |

## View the change log in the stream

```
select * from member_stream;
```

Results Data Preview

Query ID SQL 239ms 1 rows

Filter result... Copy Columns

| Row | ID | FIRST_NAME | LAST_NAME | METADATA$ACTION | METADATA$ISUPDATE | METADATA$ROW_ID |
|---|---|---|---|---|---|---|
| 1 | 2 | Anthony | Allen | DELETE | FALSE | 43c22aeb4eea86e301bd519264c9015bae0b91d6 |

## Let's execute our merge statement as shown below

```
Use CDC_STREAM;

MERGE into members_destination as T

using (select *

from member_stream
```

```
Where Not (metadata$action = 'DELETE' AND metadata$isupdate =
TRUE)) AS S

ON T.id = s.id

when matched AND S.metadata$action = 'INSERT' AND
S.metadata$isupdate

THEN

update set T.first_name = S.first_name, T.last_name =
S.last_name

When matched

And S.metadata$action = 'DELETE' THEN DELETE

when not matched

And S.metadata$action = 'INSERT' THEN

INSERT (id,

first_name,

last_name)

VALUES (S.id,

S.first_name,

S.last_name);
```

## Run merge command and let's take a look at destination table

## View the results

```
select * from members_destination;
```