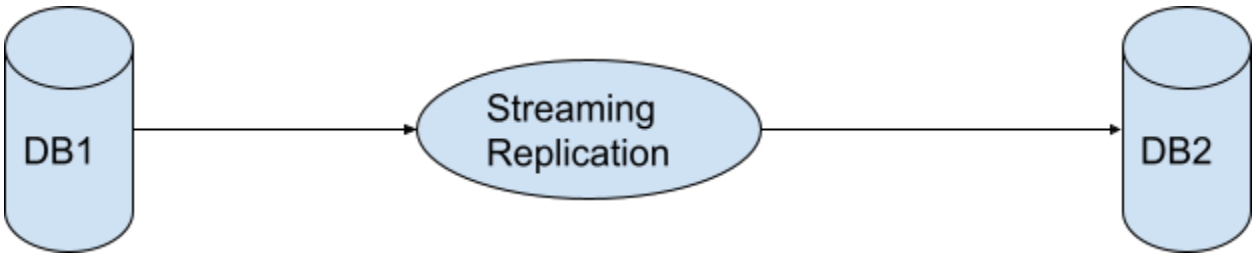
Data Tunnel

We would need to create a data tunnel to maintain a sync between 2 tables between different databases.



This steam processing would be responsible for :
Snapshot processing -> Reading all the data from DB1, and sending it asynchronously to DB2.
Stream Processing -> Once a snapshot has been synchronized, stream processing will keep looking for the new or updated data in the DB1, the moment there is any update in the data, it should send the changes to DB2.

Requirement :

Inserts/Deletes/Update => Any change on Employee_A should be reflected in Employee_B in less than <1 sec.

Approaches

Approach #1

Polling on the tables - With this approach we can create a python program which will start scanning the data from the head of the table till the last row, with every row being scanned will be sent to a streaming pipeline. This program would need to keep the track of the last offset/row which has been consumed, otherwise on the program crash this program would need to start all over again. The only problem with this approach is - this will take care of inserts, but not updates/deletes.

Approach #2

SQL Triggering (Postgres Triggers)

Using postgres triggers we can call an sql function on every row insert/update/delete.

The sql function in this case should insert all the updated rows into a new table, and then we can take the approach #1. This new table will act as a CDC table. We would also need to add the action along with the employee data. Now this action should be passed as well as with the other information to the kafka, and the consumers consuming this data will update all the syncs.

We will take approach #2 for this exercise.

Steps

1/ Modify Docker compose file to have 2 databases, both port-forwarded to different ports.

2/ Use the same below schema for employee table and add the table into both databases.

```
CREATE TABLE employees(
emp_id SERIAL,
first_name   VARCHAR(100),
last_name VARCHAR(100),
dob DATE,
city VARCHAR(100)
);
```

3/ Add PSQL function and the trigger on the employee_A which will insert the rows to the new cdc table.

```
CREATE TABLE employees_cdc(
emp_id SERIAL,
first_name   VARCHAR(100),
last_name VARCHAR(100),
dob DATE,
city VARCHAR(100),
action varchar(100)
);
```

```
4/ Modify the producer code to scan the empoloyeee_cdc table, and send the
records to the topic.

5/ Modify the consumer code, to consume the data and update the employee_B
table based on the action.
```