SQL Server To Postgres Migration

Source :-

As a source,the sql server database is hosted on EC2 instance.

A screenshot of a computer

Description automatically generated

Target :-

As a target the , Postgres database is hosted on AWS Aurora Postgres with Babelfish compatible.

A screenshot of a computer

Description automatically generated

We will be using DMS for migration of the data.

1. For one time load, we can use One time migration mode “Migrate Existing Data”
2. For full mirgration along with real time data replication we can use “Migrate Existing Data with CDC”. This will capture the changes from source and apply at target in real time.

Create DMS Endpoints :-

Endpoints for connection to source and target.

A screenshot of a computer

Description automatically generated

Create Replication Instance :-

Handles the Data load and CDC process.

A screenshot of a computer

Description automatically generated

Create DMS Tasks :-

A screenshot of a computer

Description automatically generated

Use Below JSON Code for Transformation & Selection rules:

{

"rules": [

{

"rule-type": "selection",

"rule-id": "690622224",

"rule-name": "690622224",

"object-locator": {

"schema-name": "dbo",

"table-name": "%"

},

"rule-action": "include",

"filters": []

},

{

"rule-type": "transformation",

"rule-id": "690659805",

"rule-name": "690659805",

"rule-target": "schema",

"object-locator": {

"schema-name": "dbo"

},

"rule-action": "rename",

"value": "northwind\_dbo",

"old-value": null

},

{

"rule-type": "transformation",

"rule-id": "690423845",

"rule-name": "690423845",

"rule-target": "table",

"object-locator": {

"schema-name": "%",

"table-name": "%"

},

"rule-action": "convert-lowercase",

"value": null,

"old-value": null

},

{

"rule-type": "selection",

"rule-id": "690373799",

"rule-name": "690373799",

"object-locator": {

"schema-name": "dbo",

"table-name": "aws%"

},

"rule-action": "exclude",

"filters": []

},

{

"rule-type": "validation",

"rule-id": "3",

"rule-name": "3",

"rule-target": "column",

"object-locator": {

"schema-name": "%",

"table-name": "%",

"column-name": "%",

"data-type": "datetime"

},

"rule-action": "override-validation-function",

"source-function": "case when ${column-name} is NULL then NULL else 0 end",

"target-function": "case when ${column-name} is NULL then NULL else 0 end"

}

]

}

Now Create a AWS SCT project, this will help in converting Views and Procedure definition for Postgres SQL.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Start the DMS Task :-

A screenshot of a computer

Description automatically generated

Some table might throw error , reason can be data type mismatch or non supporting data types .

Source Data Comparison :-

A screenshot of a computer

Description automatically generated

Target Data Comparison :-

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Create Remaining objects on postgres using sql ddl from AWS SCT job:-

View Recreation in Postgres:-

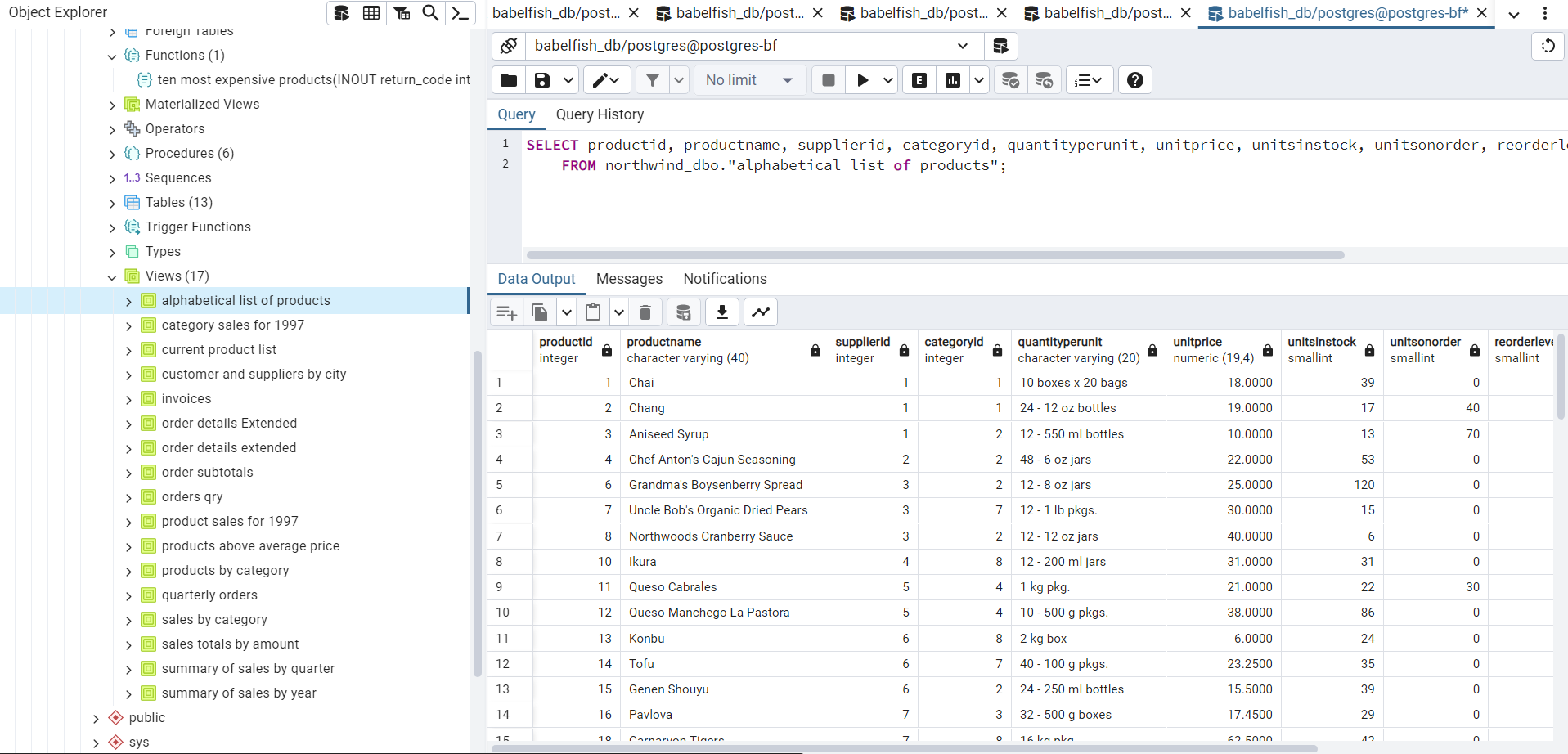
AWS SCT generated script are used to create view in Postgres.

1. Keep the generate unique names for indexes, so the view ddl will not include any random index name.

A screenshot of a computer

Description automatically generated

1. Change the ddl script to all lower case to avoid any syntax error. Use Notepad++ or any other editor.
2. Test the view once created and output can be compared with source.



Stored Procedure Recreation in Postgres :-

1. SP in postgres need to be recreated manually , if simple syntax then use traditional plpgsql language for code conversion based on the logic.
2. As shown below I have tried to convert the Stored procedure through SCT but it does not work as expected.

A screenshot of a computer

Description automatically generated

**B) Migration Strategy**

1. A client has an SQL Server database that has SSIS jobs, and a Service Broker configured on it. The database is approximately 10TB in size and grows about 10GB monthly. It currently uses 24-core VCPU and 256GB of RAM under SQL Enterprise Edition 2016. This is a transactional database that has a max downtime limit of 4 hours on special update events.
2. What will be the strategy to migrate such a database to PostGRE considering the size and transactional volume? Mention any tooling (open-source or proprietary) that can ease out this process.
3. What can be the issues being faced and possible mitigation plan?.
4. What will be the roadmap for the transition and what factors will determine the timelines of such a migration?

Solution :-

High Level Design for Migration Path :

1. Create a migration plan for the target and generate assessment reports for the migration.
2. Create a read replica for the PROD database, by AG configuration.
3. Use this replica as a source for conversion which will offload the performance impact from PROD database.
4. Configure DMS from SQL Server Replica to Postgres RDS for full load and CDC .
5. Database synchronization in real time will keep the data upto date at target side.
6. Postgres Babelfish allows to run the SQL code by maintaining the backend engine as Postgres.
7. Test the whole configuration on Non PROD env and check for challenges.