Cloud Data Warehousing: AWS RedShift

AWS Serverless and Clusters:

<https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-console-comparison.html>

1. Amazon Redshift Serverless:

a. Create a database;

b. Load the sample database table from Amazon S3;

create table users(

userid integer not null distkey sortkey,

username char(8),

firstname varchar(30),

lastname varchar(30),

city varchar(30),

state char(2),

email varchar(100),

phone char(14),

likesports boolean,

liketheatre boolean,

likeconcerts boolean,

likejazz boolean,

likeclassical boolean,

likeopera boolean,

likerock boolean,

likevegas boolean,

likebroadway boolean,

likemusicals boolean);

create table venue(

venueid smallint not null distkey sortkey,

venuename varchar(100),

venuecity varchar(30),

venuestate char(2),

venueseats integer);

create table category(

catid smallint not null distkey sortkey,

catgroup varchar(10),

catname varchar(10),

catdesc varchar(50));

create table date(

dateid smallint not null distkey sortkey,

caldate date not null,

day character(3) not null,

week smallint not null,

month character(5) not null,

qtr character(5) not null,

year smallint not null,

holiday boolean default('N'));

create table event(

eventid integer not null distkey,

venueid smallint not null,

catid smallint not null,

dateid smallint not null sortkey,

eventname varchar(200),

starttime timestamp);

create table listing(

listid integer not null distkey,

sellerid integer not null,

eventid integer not null,

dateid smallint not null sortkey,

numtickets smallint not null,

priceperticket decimal(8,2),

totalprice decimal(8,2),

listtime timestamp);

create table sales(

salesid integer not null,

listid integer not null distkey,

sellerid integer not null,

buyerid integer not null,

eventid integer not null,

dateid smallint not null sortkey,

qtysold smallint not null,

pricepaid decimal(8,2),

commission decimal(8,2),

saletime timestamp);

COPY users

FROM 's3://bilabprac/allusers\_pipe.txt'

DELIMITER '|'

TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'

IGNOREHEADER 1

REGION 'us-east-1'

IAM\_ROLE default;

COPY event

FROM 's3://bilabprac/allevents\_pipe.txt'

DELIMITER '|'

TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'

IGNOREHEADER 1

REGION 'us-east-1'

IAM\_ROLE default;

COPY sales

FROM 's3://bilabprac/sales\_tab.txt'

DELIMITER '\t'

TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'

IGNOREHEADER 1

REGION 'us-east-1'

IAM\_ROLE default;

c. Try sample queries;

SELECT sum(qtysold)

FROM sales, date

WHERE sales.dateid = date.dateid

AND caldate = '2008-01-05';

-- Find top 10 buyers by quantity.

SELECT firstname, lastname, total\_quantity

FROM (SELECT buyerid, sum(qtysold) total\_quantity

FROM sales

GROUP BY buyerid

ORDER BY total\_quantity desc limit 10) Q, users

WHERE Q.buyerid = userid

ORDER BY Q.total\_quantity desc;

-- Find events in the 99.9 percentile in terms of all time gross sales.

SELECT eventname, total\_price

FROM (SELECT eventid, total\_price, ntile(1000) over(order by total\_price desc) as percentile

FROM (SELECT eventid, sum(pricepaid) total\_price

FROM sales

GROUP BY eventid)) Q, event E

WHERE Q.eventid = E.eventid

AND percentile = 1

ORDER BY total\_price desc;

2. Amazon Redshift Cluster:

a. Create a cluster;

b. Create a database;

c. Load the sample database table from Amazon S3;

d. Try sample queries;

3. Sort Key and Distribution style

Amazon Redshift’s DISTKEY and SORTKEY are powerful tools for optimizing query performance. Because Redshift is a columnar database with compressed storage, it doesn't use indexes like transactional databases — such as MySQL, Microsoft SQL, and PostgreSQL — would. Instead, it uses DISTKEYs and SORTKEYs.

SORTKEY

Keyword that specifies that the column is the sort key for the table. When data is loaded into the table, the data is sorted by one or more columns that are designated as sort keys. You can use the SORTKEY keyword after a column name to specify a single-column sort key, or you can specify one or more columns as sort key columns for the table by using the SORTKEY (column\_name [, ...]) syntax. Only compound sort keys are created with this syntax.

If you don't specify any sort keys, the table isn't sorted. You can define a maximum of 400 SORTKEY columns per table.

For example:

create table sales(

salesid integer not null,

listid integer not null distkey,

eventid integer not null sortkey,

dateid smallint not null sortkey,

);

For more info: <https://docs.aws.amazon.com/redshift/latest/dg/tutorial-tuning-tables-sort-keys.html>

**DIST Key**

Redshift Distribution Keys (DIST Keys) determine where data is stored in Redshift. Clusters store data fundamentally across the compute nodes. Query performance suffers when a large amount of data is stored on a single node.

**Types of Distribution Styles:**

When you create a table, you can designate one of four distribution styles; AUTO, EVEN, KEY, or ALL.

Even Distribution

This is the default distribution styles of a table. In Even Distribution the Leader node of the cluster distributes the data of a table evenly across all slices, using a round-robin approach.

Key Distribution

The data is distributed across slices by the leader node matching the values of a designated column. So all the entries with the same value in the column end up in the same slice.

All Distribution

Leader node maintains a copy of the table on all the computing nodes resulting in more space utilization. Since all the nodes have a local copy of the data, the query does not require copying data across the network. This results in *faster query* operations. The negative side of using ALL is that a copy of the table is on every node in the cluster. This takes up *too much of space* and increases the time taken by *Copy command* to upload data into Redshift.

AUTO distribution

With AUTO distribution, Amazon Redshift assigns an optimal distribution style based on the size of the table data. For example, if AUTO distribution style is specified, Amazon Redshift initially assigns the ALL distribution style to a small table. When the table grows larger, Amazon Redshift might change the distribution style to KEY, choosing the primary key (or a column of the composite primary key) as the distribution key. If the table grows larger and none of the columns are suitable to be the distribution key, Amazon Redshift changes the distribution style to EVEN. The change in distribution style occurs in the background with minimal impact to user queries.

**Choosing the Right Distribution Styles**

Choose columns used in the query that leads to least skewness as the DISTKEY.

a. If the table(e.g. fact table) is highly deformalized and no JOIN is required, choose the EVEN style.

b. Choose ALL style for small tables that do not often change. For example, a table containing telephone ISD codes against the country name.

c. It is beneficial to select a KEY distribution if a table is used in JOINS. Also, consider the other joining tables and their distribution style.

d. If one particular node contains the skew data, the processing on this node will be slower. This results in much longer total query processing time. This query under skewed configuration may take even longer than the query made against the table without a DISTKEY.

For more information: <https://docs.aws.amazon.com/redshift/latest/dg/t_Distributing_data.html>

4. **EXPLAIN** function (calculating the cost of a query):

Displays the execution plan for a query statement without running the query.

Syntax:

EXPLAIN [ VERBOSE ] query

The following example returns the query plan for a query that selects the EVENTID, EVENTNAME, VENUEID, and VENUENAME from the EVENT and VENUE tables:

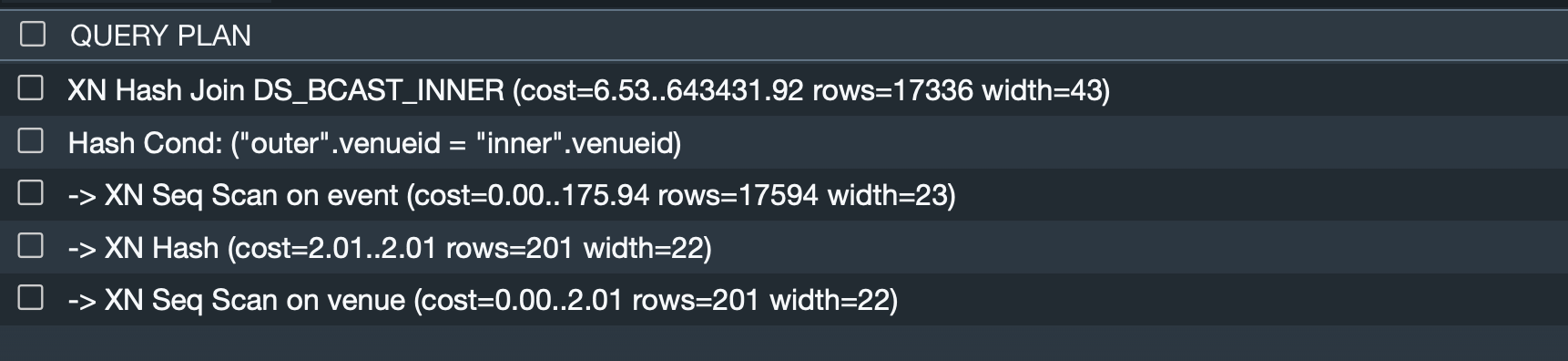
explain

select eventid, eventname, event.venueid, venuename

from event, venue

where event.venueid = venue.venueid;

Outputs:



5. Connect Tableau with Amazon RedShift:

(1). Make the connection and set up the data source

a. Start Tableau and under Connect, select Amazon Redshift. For a complete list of data connections, select More under To a Server. Then do the following:

b. Enter the name of the server that hosts the database and the name of the database you want to connect to.

c. Enter the user name and password. Tick the Require SSL box when connecting to an SSL server (change the inbound rules).

d. (Optional) Select Initial SQL to specify a SQL command to run at the beginning of every connection, such as when you open the workbook, refresh an extract, sign in to Tableau Server or publish to Tableau Server.

e. Select Sign In.

If Tableau can't make the connection, verify that your credentials are correct. If you still can't connect, your computer is having trouble locating the server. Contact your network administrator or database administrator.

(2). On the data source page, do the following:

a. Select the default data source name at the top of the page, and then enter a unique data source name for use in Tableau. For example, use a data source naming convention that helps other users of the data source figure out which data source to connect to.

b. From the Schema drop-down list, select a schema or use the text box to search for a schema by name.

c. Under Table, select a table or use the text box to search for a table by name.

d. Drag the table to the canvas, and then select the sheet tab to start your analysis.

Use custom SQL to connect to a specific query rather than the entire data source.

For more info : <https://help.tableau.com/current/pro/desktop/en-gb/examples_amazonredshift.htm>