L1 E3 - Columnar Vs Row Storage

July 1, 2021

1 Exercise 03 - Columnar Vs Row Storage

- The columnar storage extension used here:
  - cstore_fdw by citus_data https://github.com/citusdata/cstore_fdw
- The data tables are the ones used by citus_data to show the storage extension

```
In [1]: %load_ext sql
```

1.1 STEP 0 : Connect to the local database where Pagila is loaded

1.1.1 Create the database

```
In [2]: !sudo -u postgres psql -c 'CREATE DATABASE reviews;'

        !wget http://examples.citusdata.com/customer_reviews_1998.csv.gz
        !wget http://examples.citusdata.com/customer_reviews_1999.csv.gz

        !gzip -d customer_reviews_1998.csv.gz
        !gzip -d customer_reviews_1999.csv.gz

        !mv customer_reviews_1998.csv /tmp/customer_reviews_1998.csv
        !mv customer_reviews_1999.csv /tmp/customer_reviews_1999.csv
```

```
CREATE DATABASE
--2021-07-01 23:19:30--  http://examples.citusdata.com/customer_reviews_1998.csv.gz
Resolving examples.citusdata.com (examples.citusdata.com)... 104.26.15.56, 172.67.73.2, 104.26.1
Connecting to examples.citusdata.com (examples.citusdata.com)|104.26.15.56|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://examples.citusdata.com/customer_reviews_1998.csv.gz [following]
--2021-07-01 23:19:30--  https://examples.citusdata.com/customer_reviews_1998.csv.gz
Connecting to examples.citusdata.com (examples.citusdata.com)|104.26.15.56|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24774482 (24M) [application/x-gzip]
Saving to: customer_reviews_1998.csv.gz

customer_reviews_19 100%[===================>]  23.63M  49.6MB/s    in 0.5s
```

```
2021-07-01 23:19:31 (49.6 MB/s) - customer_reviews_1998.csv.gz saved [24774482/24774482]

URL transformed to HTTPS due to an HSTS policy
--2021-07-01 23:19:32--  https://examples.citusdata.com/customer_reviews_1999.csv.gz
Resolving examples.citusdata.com (examples.citusdata.com)... 104.26.15.56, 172.67.73.2, 104.26.1
Connecting to examples.citusdata.com (examples.citusdata.com)|104.26.15.56|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48996256 (47M) [application/x-gzip]
Saving to: customer_reviews_1999.csv.gz

customer_reviews_19 100%[===================>]  46.73M  35.4MB/s    in 1.3s

2021-07-01 23:19:34 (35.4 MB/s) - customer_reviews_1999.csv.gz saved [48996256/48996256]
```

1.1.2 Connect to the database

```
In [6]: DB_ENDPOINT = "127.0.0.1"
        DB = 'reviews'
        DB_USER = 'student'
        DB_PASSWORD = 'student'
        DB_PORT = '5432'

        # postgresql://username:password@host:port/database
        conn_string = "postgresql://{}:{}@{}:{}/{}" \
                                .format(DB_USER, DB_PASSWORD, DB_ENDPOINT, DB_PORT, DB)

        print(conn_string)
```

```
postgresql://student:student@127.0.0.1:5432/reviews
```

```
In [7]: %sql $conn_string
```

```
Out[7]: 'Connected: student@reviews'
```

1.2 STEP 1 : Create a table with a normal (Row) storage & load data

TODO: Create a table called customer_reviews_row with the column names contained in the `customer_reviews_1998.csv` and `customer_reviews_1999.csv` files.

```
In [9]: %%sql
        DROP TABLE IF EXISTS customer_reviews_row;
        CREATE TABLE customer_reviews_row
        (
            customer_id TEXT,
            review_date DATE,
            review_rating INTEGER,
```

```
            review_votes INTEGER,
            review_helpful_votes INTEGER,
            product_id CHAR(10),
            product_title TEXT,
            product_sales_rank BIGINT,
            product_group TEXT,
            product_category TEXT,
            product_subcategory TEXT,
            similar_product_ids CHAR(10)[]
        )
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
Done.
Done.
```

```
Out[9]: []
```

TODO: Use the COPY statement to populate the tables with the data in the `customer_reviews_1998.csv` and `customer_reviews_1999.csv` files. You can access the files in the `/tmp/` folder.

```
In [10]: %%sql
         COPY customer_reviews_row FROM '/tmp/customer_reviews_1998.csv' WITH CSV;
         COPY customer_reviews_row FROM '/tmp/customer_reviews_1999.csv' WITH CSV;
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
589859 rows affected.
1172645 rows affected.
```

```
Out[10]: []
```

1.3 STEP 2 : Create a table with columnar storage & load data

First, load the extension to use columnar storage in Postgres.

```
In [11]: %%sql

         -- load extension first time after install
         CREATE EXTENSION cstore_fdw;

         -- create server object
         CREATE SERVER cstore_server FOREIGN DATA WRAPPER cstore_fdw;
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
Done.
Done.
```

Out[11]: []

TODO: Create a `FOREIGN TABLE` called `customer_reviews_col` with the column names contained in the `customer_reviews_1998.csv` and `customer_reviews_1999.csv` files.

In [12]:
```
%%sql
-- create foreign table
DROP FOREIGN TABLE IF EXISTS customer_reviews_col;

-------------
CREATE FOREIGN TABLE customer_reviews_col
(
    customer_id TEXT,
    review_date DATE,
    review_rating INTEGER,
    review_votes INTEGER,
    review_helpful_votes INTEGER,
    product_id CHAR(10),
    product_title TEXT,
    product_sales_rank BIGINT,
    product_group TEXT,
    product_category TEXT,
    product_subcategory TEXT,
    similar_product_ids CHAR(10)[]
)

-------------
-- leave code below as is
SERVER cstore_server
OPTIONS(compression 'pglz');
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
Done.
Done.
```

Out[12]: []

TODO: Use the COPY statement to populate the tables with the data in the `customer_reviews_1998.csv` and `customer_reviews_1999.csv` files. You can access the files in the `/tmp/` folder.

In [13]:
```
%%sql
COPY customer_reviews_col FROM '/tmp/customer_reviews_1998.csv' WITH CSV;
COPY customer_reviews_col FROM '/tmp/customer_reviews_1999.csv' WITH CSV;
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
589859 rows affected.
1172645 rows affected.
```

Out[13]: []

1.4 Step 3: Compare perfromamce

Now run the same query on the two tables and compare the run time. Which form of storage is more performant?

TODO: Write a query that calculates the average `review_rating` by `product_title` for all reviews in 1995. Sort the data by `review_rating` in descending order. Limit the results to 20.

First run the query on `customer_reviews_row`:

```
In [18]: %%time
         %%sql

         SELECT product_title, avg(review_rating)
         FROM customer_reviews_row
         where extract(YEAR FROM review_date) = 1995
         group by 1
         order by 2 desc
         limit 20
         ;
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
20 rows affected.
CPU times: user 4.49 ms, sys: 0 ns, total: 4.49 ms
Wall time: 560 ms
```

```
Out[18]: [('Beer Games 2, Revised ', Decimal('5.0000000000000000')),
          ('Making the Most of Your Money', Decimal('5.0000000000000000')),
          ('Griffin and Sabine', Decimal('5.0000000000000000')),
          ("A People's History of the United States", Decimal('5.0000000000000000')),
          ('The Joy Luck Club ', Decimal('5.0000000000000000')),
          ('Insomnia', Decimal('5.0000000000000000')),
          ('Interview with the Vampire', Decimal('5.0000000000000000')),
          ('The Name of the Rose', Decimal('5.0000000000000000')),
          ('Snow Crash (Bantam Spectra Book)', Decimal('5.0000000000000000')),
          ('Lizard King', Decimal('5.0000000000000000')),
          ('Amber Diceless Role-Playing', Decimal('5.0000000000000000')),
          ('Radical Honesty ', Decimal('5.0000000000000000')),
          ('The C++ Programming Language (3rd Edition)', Decimal('5.0000000000000000')),
          ('Hatchet (Large Print Cornerstone Ser)', Decimal('5.0000000000000000')),
          ('The Long Goodbye (Vintage Crime/Black Lizard)', Decimal('5.0000000000000000')),
          ('Virus of the Mind', Decimal('5.0000000000000000')),
          ('Simulating Neural Networks With Mathematica', Decimal('5.0000000000000000')),
          ("Wizard's First Rule (Sword Of Truth)", Decimal('5.0000000000000000')),
          ('Hula', Decimal('5.0000000000000000')),
          ('The Illuminatus! Trilogy ', Decimal('5.0000000000000000'))]
```

Then on `customer_reviews_col`:

```
In [19]: %%time
         %%sql
```

```
        SELECT product_title, avg(review_rating)
        FROM customer_reviews_col
        where extract(YEAR FROM review_date) = 1995
        group by 1
        order by 2 desc
        limit 20
        ;
```

```
 * postgresql://student:***@127.0.0.1:5432/reviews
20 rows affected.
CPU times: user 4.19 ms, sys: 0 ns, total: 4.19 ms
Wall time: 585 ms
```

```
Out[19]: [('Beer Games 2, Revised ', Decimal('5.0000000000000000')),
          ('Making the Most of Your Money', Decimal('5.0000000000000000')),
          ('Griffin and Sabine', Decimal('5.0000000000000000')),
          ("A People's History of the United States", Decimal('5.0000000000000000')),
          ('The Joy Luck Club ', Decimal('5.0000000000000000')),
          ('Insomnia', Decimal('5.0000000000000000')),
          ('Interview with the Vampire', Decimal('5.0000000000000000')),
          ('The Name of the Rose', Decimal('5.0000000000000000')),
          ('Snow Crash (Bantam Spectra Book)', Decimal('5.0000000000000000')),
          ('Lizard King', Decimal('5.0000000000000000')),
          ('Amber Diceless Role-Playing', Decimal('5.0000000000000000')),
          ('Radical Honesty ', Decimal('5.0000000000000000')),
          ('The C++ Programming Language (3rd Edition)', Decimal('5.0000000000000000')),
          ('Hatchet (Large Print Cornerstone Ser)', Decimal('5.0000000000000000')),
          ('The Long Goodbye (Vintage Crime/Black Lizard)', Decimal('5.0000000000000000')),
          ('Virus of the Mind', Decimal('5.0000000000000000')),
          ('Simulating Neural Networks With Mathematica', Decimal('5.0000000000000000')),
          ("Wizard's First Rule (Sword Of Truth)", Decimal('5.0000000000000000')),
          ('Hula', Decimal('5.0000000000000000')),
          ('The Illuminatus! Trilogy ', Decimal('5.0000000000000000'))]
```

1.5 Conclusion: We can see that the columnar storage is faster!