Loading Data into Amazon Redshift from Amazon S3

By Danijel Hafner, BI Developer, iOLAP, Inc.

Objective

Amazon offers loading data to Redshift either from flat files that are stored in an Amazon S3 bucket or from Amazon DynamoDB table. In this article, I will show you how to load data from your local machine to Amazon Redshift using the Amazon S3 service. Also, for inserting data to Amazon Redshift, I will show you how to use COPY command.

Test Overview

For this test, I used Part Table with 20 million rows and a size of 2.83 GB. I loaded data first from my local machine to the S3 bucket and then from Amazon S3 into Amazon Redshift. You can find Part Table in Amazon S3 bucket: s3://redshift-demo/tpc-h/100/part/part.tbl.

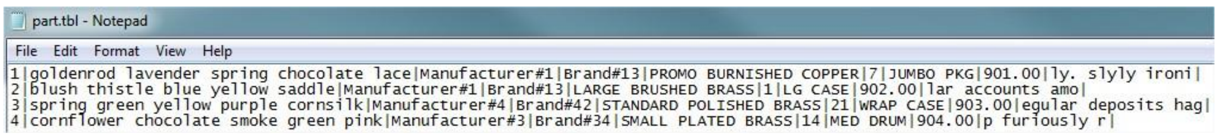
I used an Amazon Redshift cluster with one extra-large node (XL) with 2TB of compressed storage. This storage amount much more than required for my test but it comes in a minimum one-node package. Listed below is the XL node specification.

<u>High Storage Extra Large (XL) DW Node:</u>
- CPU: 2 virtual cores - Intel Xeon E5
- ECU: 4.4
- Memory: 15 GiB
- Storage: 3 HDD with 2TB of local attached storage

Preparing Data for Loading

There is a couple of things we can do to speed up loading process.

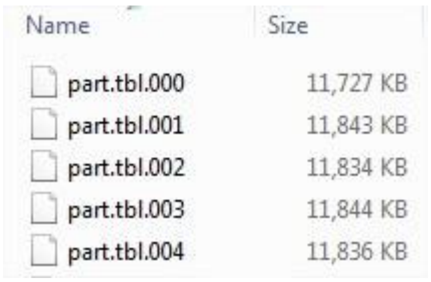
<u>Step 1</u>
Store your data in sort order so you don't need to run a VACUUM command. For example, the timestamp column can be a good sort key example when we load files on a daily bases. Also, the delimiter can be the last field in a record.



```
1|goldenrod lavender spring chocolate lace|Manufacturer#1|Brand#13|PROMO BURNISHED COPPER|7|JUMBO PKG|901.00|ly. slyly ironi|
2|blush thistle blue yellow saddle|Manufacturer#1|Brand#13|LARGE BRUSHED BRASS|1|LG CASE|902.00|lar accounts amo|
3|spring green yellow purple cornsilk|Manufacturer#4|Brand#42|STANDARD POLISHED BRASS|21|WRAP CASE|903.00|egular deposits hag|
4|cornflower chocolate smoke green pink|Manufacturer#3|Brand#34|SMALL PLATED BRASS|14|MED DRUM|904.00|p furiously r|
```

Step 2

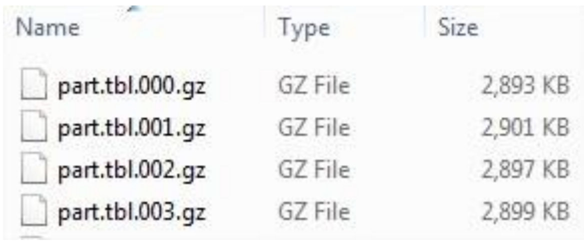
Redshift uses a massively parallel processing (MPP) architecture to read and load data in parallel from the S3 bucket so I want to split my large flat file into multiple files to take advantage of parallelism. The COPY command loads the data in parallel from multiple files, dividing the workload among the nodes in the cluster. The number of files should correspond to the number of slices in the cluster. I used one XL compute node with two slices and divided the file into 200 files. All files should be roughly the same size and must share a common prefix for the set.



Name	Size
part.tbl.000	11,727 KB
part.tbl.001	11,843 KB
part.tbl.002	11,834 KB
part.tbl.003	11,844 KB
part.tbl.004	11,836 KB

Step 3

Compress the files individually using GZIP or LZOP. I used GZIP and reduced size from 2.83GB to 567MB. This is only file compression. However, Amazon Redshift also has its own column compression methods once the data comes in.

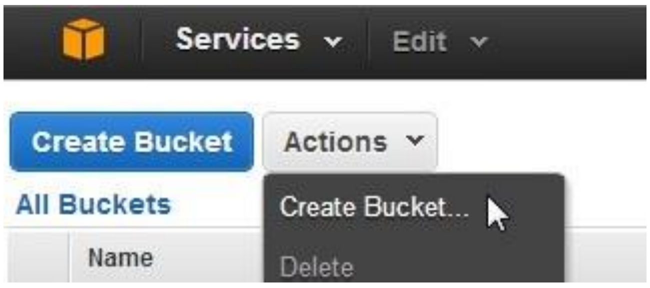


Name	Type	Size
part.tbl.000.gz	GZ File	2,893 KB
part.tbl.001.gz	GZ File	2,901 KB
part.tbl.002.gz	GZ File	2,897 KB
part.tbl.003.gz	GZ File	2,899 KB

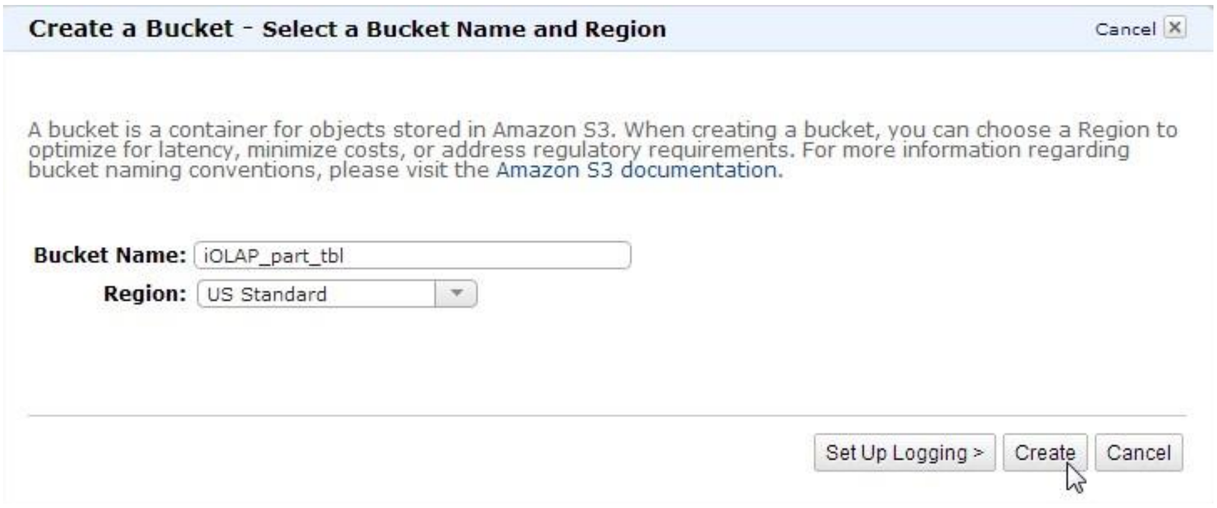
After this step, I can upload files to the Amazon S3 bucket.

Upload to Amazon S3

For loading to the Amazon S3 bucket I used the S3 console. This Amazon service is used to store and retrieve any amount of data and works very well. It is simple, easy to use and worked without any problems for my tests. This task can be performed with some other tools like S3 browser, Bucket Explorer, S3 Browser for Chrome and so on.



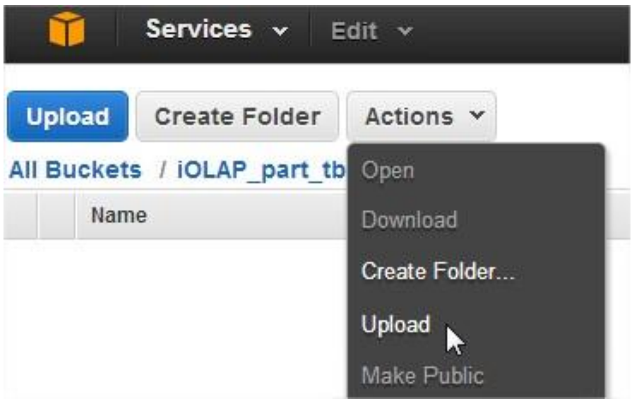
I created "bucket" named iOLAP_part_tbl.


Create a Bucket - Select a Bucket Name and Region

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

Bucket Name: iOLAP_part_tbl
Region: US Standard

Once I have my bucket, I upload all Part files from my computer.



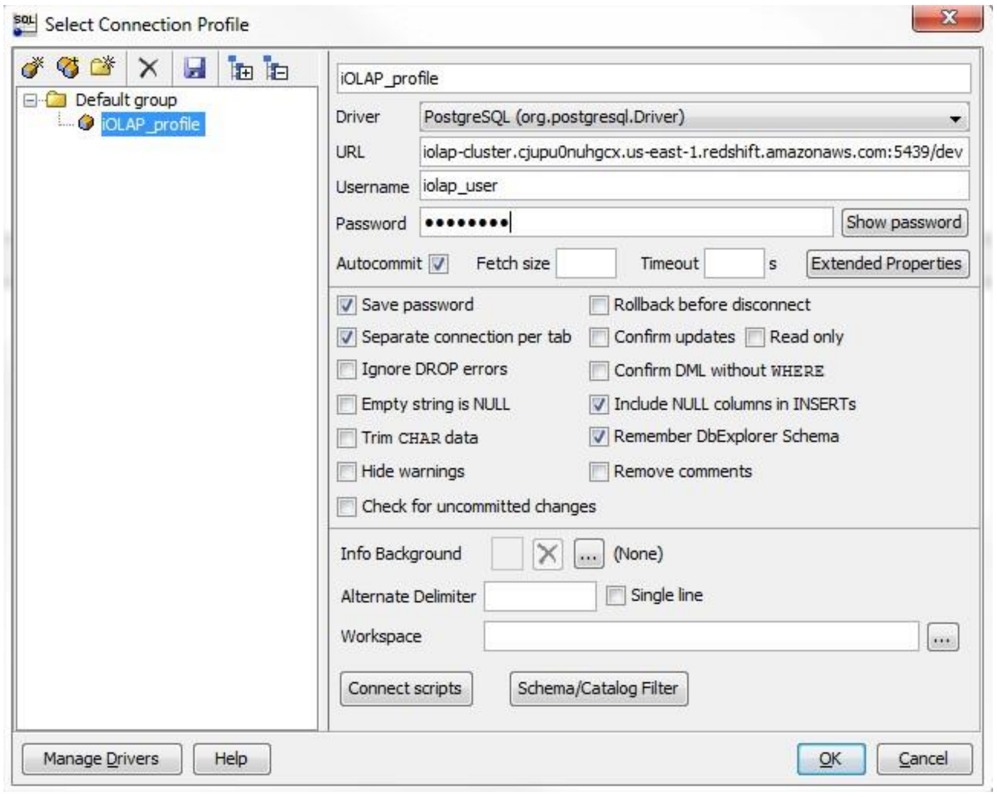
For 567MB of 200 files, the load took 7min 23sec. Upload speed was around 1300 KB/sec.

Loading Redshift Data from Amazon S3

I used the COPY command to load data to Redshift. It is the best way to load large amounts of data from Amazon S3 (or DynamoDB). The COPY command loads data in parallel to each compute node. Before loading, I needed to have my target table already created. To do this I must connect to my cluster. I used SQL Workbench/J, a free SQL client tool. My workbench needed a driver that will enable connections to the cluster. Amazon Redshift supports the following version 8 JDBC and ODBC drivers:

- JDBC http://jdbc.postgresql.org/download/postgresql-8.4-703.jdbc4.jar
- ODBC (versions below or later)
  - 32-bit http://ftp.postgresql.org/pub/odbc/versions/msi/psqlodbc_08_04_0200.zip
  - 64-bit http://ftp.postgresql.org/pub/odbc/versions/msi/psqlodbc_09_00_0101-x64.zip

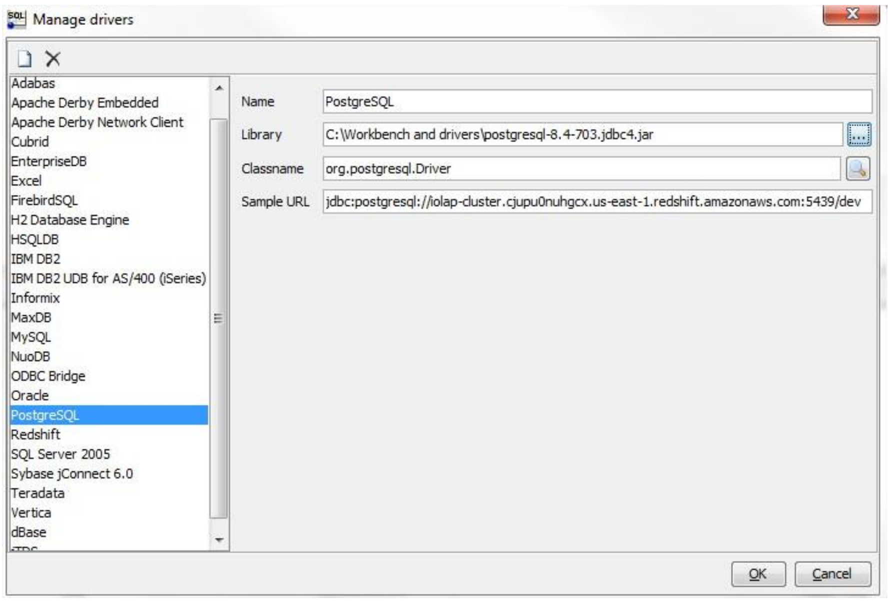
Below are the SQL Workbench/J setup connection properties:



Profile name: iOLAP_profile
Driver: PostgreSQL (org.postgresql.Driver)
URL: iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com:5439/dev
Username: iolap_user
Password: ••••••••
Autocommit ✓ Fetch size Timeout s

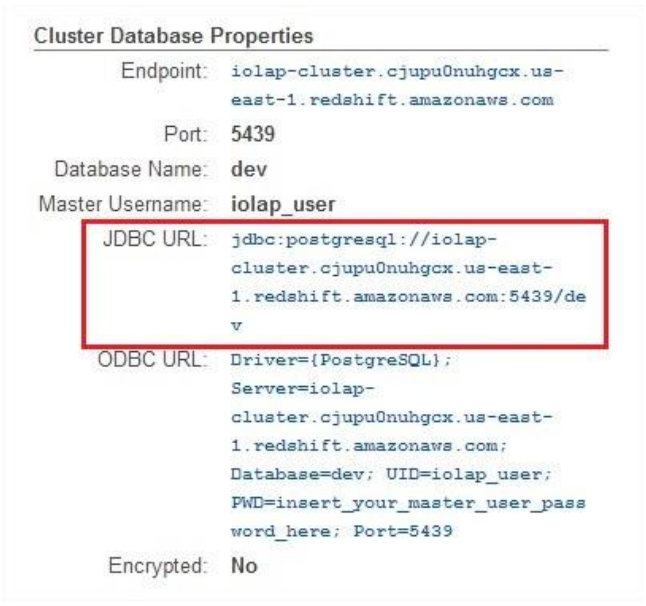
- ✓ Save password
- ✓ Separate connection per tab
- ☐ Ignore DROP errors
- ☐ Empty string is NULL
- ☐ Trim CHAR data
- ☐ Hide warnings
- ☐ Check for uncommitted changes
- ☐ Rollback before disconnect
- ☐ Confirm updates
- ☐ Read only
- ☐ Confirm DML without WHERE
- ✓ Include NULL columns in INSERTs
- ✓ Remember DbExplorer Schema
- ☐ Remove comments

Info Background (None)
Alternate Delimiter ☐ Single line
Workspace



Name: PostgreSQL
Library: C:\Workbench and drivers\postgresql-8.4-703.jdbc4.jar
Classname: org.postgresql.Driver
Sample URL: jdbc:postgresql://iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com:5439/dev

The URL is located in the cluster properties in AWS console:



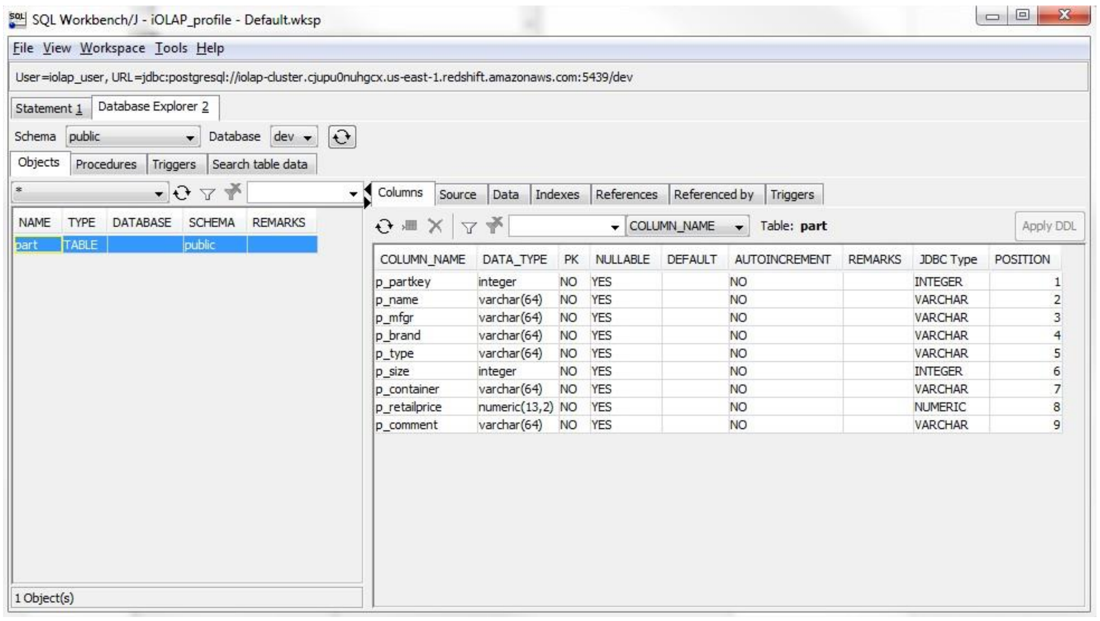
Cluster Database Properties

Endpoint: iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com
Port: 5439
Database Name: dev
Master Username: iolap_user
JDBC URL: jdbc:postgresql://iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com:5439/dev
ODBC URL: Driver={PostgreSQL}; Server=iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com; Database=dev; UID=iolap_user; PWD=insert_your_master_user_password_here; Port=5439
Encrypted: No

Before loading I created the target table:

```sql
CREATE TABLE part(
        P_PartKey int ,
        P_Name varchar(64) ,
        P_Mfgr varchar(64) ,
        P_Brand varchar(64) ,
        P_Type varchar(64) ,
        P_Size int ,
        P_Container varchar(64) ,
        P_RetailPrice decimal(13, 2) ,
        P_Comment varchar(64)
);
```

Database explorer in SQL Workbench/J:



User=iolap_user, URL=jdbc:postgresql://iolap-cluster.cjupu0nuhgcx.us-east-1.redshift.amazonaws.com:5439/dev

Schema: public Database: dev

NAME	TYPE	DATABASE	SCHEMA	REMARKS
part	TABLE		public	

COLUMN_NAME Table: **part**

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DEFAULT	AUTOINCREMENT	REMARKS	JDBC Type	POSITION
p_partkey	integer	NO	YES		NO		INTEGER	1
p_name	varchar(64)	NO	YES		NO		VARCHAR	2
p_mfgr	varchar(64)	NO	YES		NO		VARCHAR	3
p_brand	varchar(64)	NO	YES		NO		VARCHAR	4
p_type	varchar(64)	NO	YES		NO		VARCHAR	5
p_size	integer	NO	YES		NO		INTEGER	6
p_container	varchar(64)	NO	YES		NO		VARCHAR	7
p_retailprice	numeric(13,2)	NO	YES		NO		NUMERIC	8
p_comment	varchar(64)	NO	YES		NO		VARCHAR	9

Syntax of the COPY command:

```
copy <table_name> from 's3://<bucket_name>/<object_prefix>'
CREDENTIALS 'aws_access_key_id=<my aws_access key id>;
aws_secret_access_key=<my aws secret access key>' [<options>];
```

My COPY command for loading the Part table:

```
copy part from 's3://iOLAP_part_tbl/part.tbl' CREDENTIALS
'aws_access_key_id={ID};aws_secret_access_key={key}' delimiter '|' gzip;
```

<u>Statistics for loading 20 million records with one XL node:</u>
Execution time: 2 minutes and 28 seconds
Rows per second: 135,000
MB per second: 15

This result is very acceptable for a cluster configuration with only one node. Amazon Redshift allows you to scale a XL cluster up to 32 nodes and 8XL clusters up to 100 nodes. The upload speed is not proportional to the number of nodes.

Two system tables can be helpful in troubleshooting data load issues:

- STL_LOAD_ERRORS
- STL_FILE_SCAN

Unload Data

Like the COPY command for loading data, Amazon Redshift has a command to unload data from Amazon Redshift to Amazon S3. This command will copy data back to Amazon S3 with the number of files equal to the number of slices in the current cluster. Amazon Redshift offers some options for this command like compress to GZIP or choosing delimiter type. The original data will remain in Amazon Redshift.

```
unload ('SELECT * FROM part')   to 's3://part_unload/'
CREDENTIALS 'aws_access_key_id={ID};aws_secret_access_key={key}' delimiter
'|' gzip;
```

Execution time was 2 minutes 25 seconds.

Summary

The COPY command is simple and very fast once the data is up on Amazon servers. Potential bottlenecks for large data volumes can be uploaded to Amazon S3. I used a cluster with the minimum number of nodes and loading times were very good. More nodes in cluster will result in better performance and it's important to know how many nodes you need based on your business requirements.

About the Author

Danijel Hafner is a BI Developer at iOLAP d.o.o (a subsidiary of iOLAP Inc.) located in Rijeka, Croatia. He graduated from Faculty of Organization and Informatics in Varaždin, Croatia. Danijel has been with iOLAP since 2010 and has worked on several highly visible projects in the U.S. He specializes in Data Warehousing and ETL development.