Intro to Redshift

Jason DeRose

What is Redshift?

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse that makes it simple and cost-effective to analyze all your data using your existing business intelligence tools.

What is Redshift?

- A fork of (a really old version of) PostgreSQL (with many features disabled)
- Traditional table-based relational database
- Query with SQL via ODBC/JDBC drivers
- Server managed by AWS (similar to RDS)
- Column-oriented data storage
- Parallel processing of queries

When to use Redshift instead of RDS

Redshift / Data Warehouses	- VS -	General purpose SQL databases
Ideal for OLAP (online analytical processing) of structured data		Ideal for OLTP (online transaction processing) of structured data
Large number of READS only, aggregating large volumes of data		Large number of READS and WRITES on individual records
Typically used internally (example: reporting historical monthly sales figures per product category)		Typically used by end-customer (example: placing an order for a product)

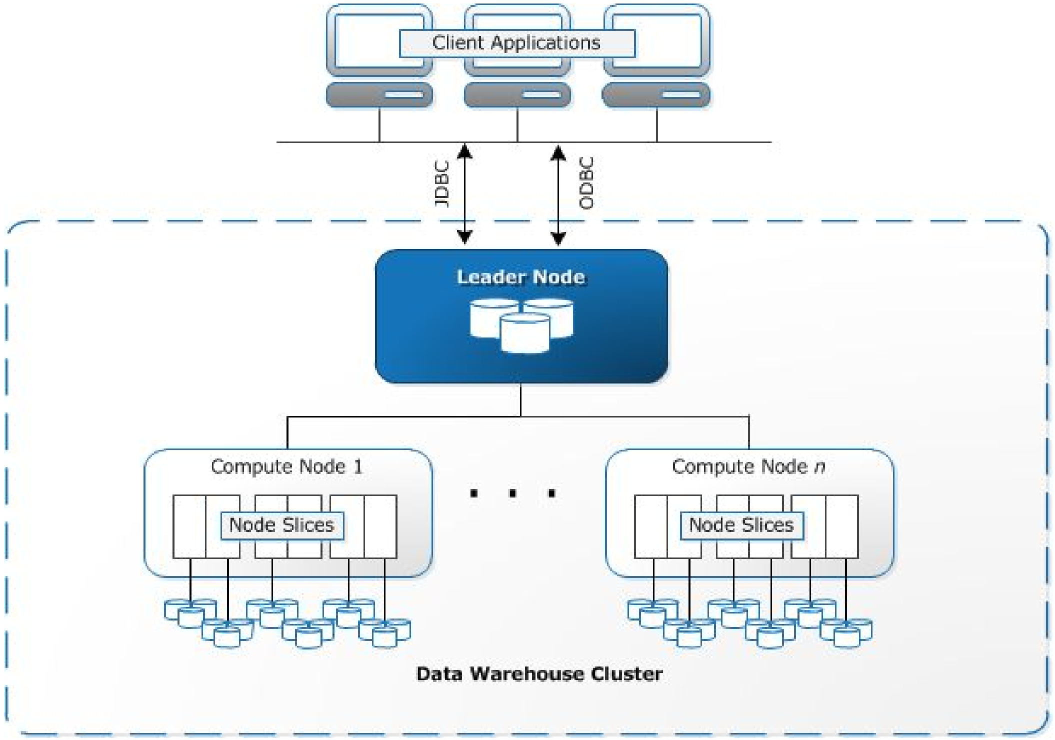
Parallel processing

Leader Node

- Receives incoming queries
- Derives execution plan
- Merges results from compute clusters

Compute Node

- Has own dedicated CPU, memory, and storage
- Returns data it stores that belongs in the query result
- Stores redundant data from other nodes



Current Pricing

Dense Compute Cluster

Node Size	vCPU	ECU	RAM (GiB)	Slices Per Node	Storage Per Node	N. VA Price Per Node	Node Range	Total Capacity
dc1.large	2	7	15	2	160 GB SSD	$ 0.25	1–32	5.12 TB
dc1.8xlarge	32	104	244	32	2.56 TB SSD	$ 4.80	2–128	326 TB

Dense Storage Cluster

Node Size	vCPU	ECU	RAM (GiB)	Slices Per Node	Storage Per Node	N. VA Price Per Node	Node Range	Total Capacity
ds2.xlarge	4	13	31	2	2 TB HDD	$ 0.85	1–32	64 TB
ds2.8xlarge	36	119	244	16	16 TB HDD	$ 6.80	2–128	2 PB

Reserved instance pricing available

Loading data (COPY)

The COPY command loads data in bulk in CSV, Avro or JSON from

- S3
- EMR
- Custom SSH command
- Or, directly from DynamoDB tables

Data can also be unloaded from other services (ie. kinesis firehose) to S3 and then COPY into Redshift

COPY can be automated from specific S3 location w/ Lambda

Loading data (INSERT)

Individual INSERT statements are slow. Multi-row or bulk inserts should be used instead.

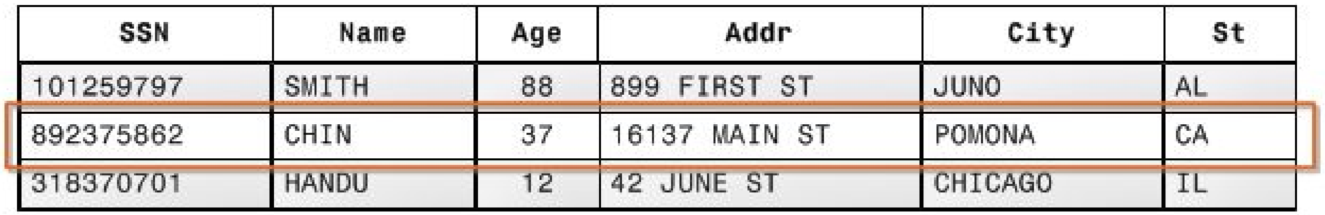
Multi-row INSERT

```
insert into category_stage values
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);
```

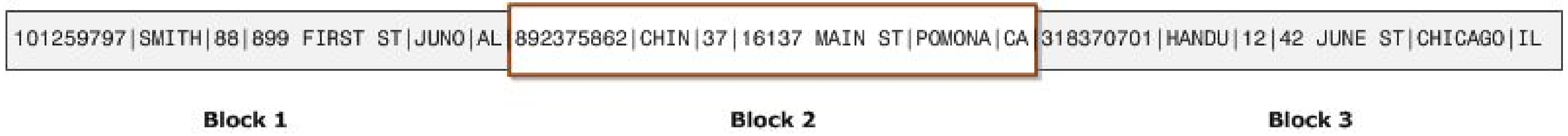
Bulk INSERT

```
insert into category_stage
(select * from category);
```

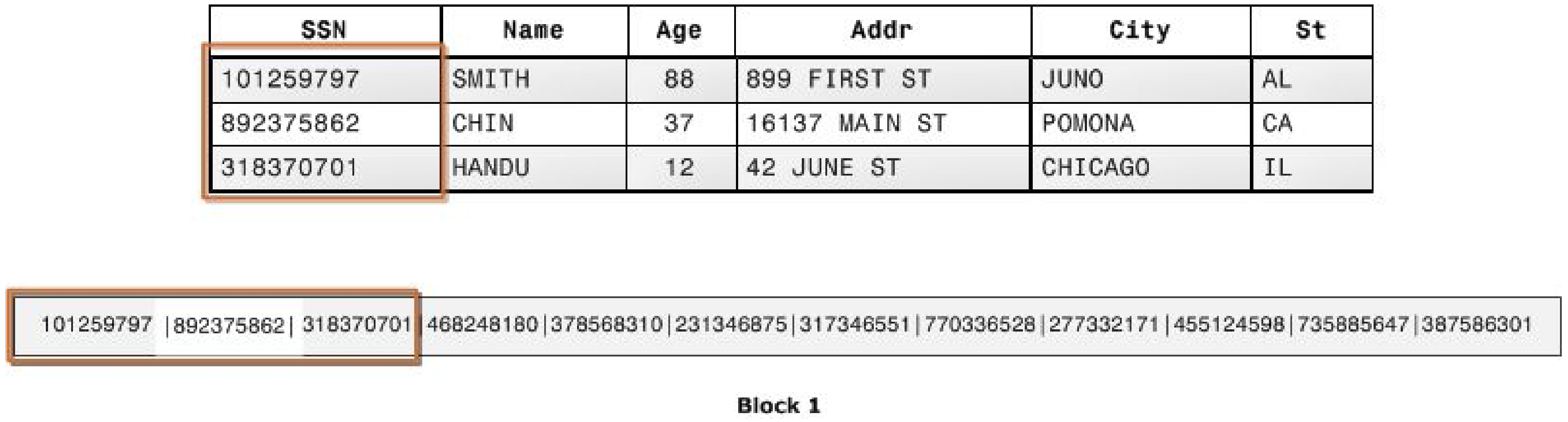
Row-oriented storage (traditional database)



SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL


101259797|SMITH|88|899 FIRST ST|JUNO|AL|892375862|CHIN|37|16137 MAIN ST|POMONA|CA|318370701|HANDU|12|42 JUNE ST|CHICAGO|IL
Block 1
Block 2
Block 3

Column-oriented storage (Redshift, analytics-focused)



SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

```
101259797 |892375862| 318370701|468248180|378568310|231346875|317346551|770336528|277332171|455124598|735885647|387586301
```

Block 1

Compression

1. **Reduces disk usage**
2. **Reduces I/O (improving query performance)**

Types (set per column)

- Raw
- Byte-Dictionary
- Delta
- LZO
- Mostly
- Runlength
- Text255 and Text32k
- Zstandard

Example: Run-Length Encoding

Blue,Blue,Blue,Blue,Blue,Blue,Green,Green,Green

$\downarrow$

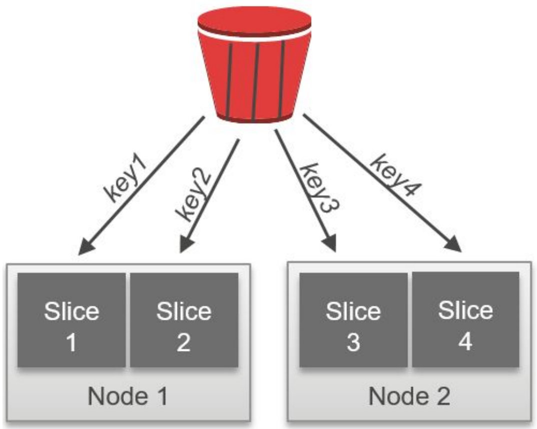
{6,Blue},{3,Green}

Compression type chosen automatically when empty table populated via COPY. Otherwise, ANALYZE COMPRESSION will suggest best choices.

Table Distribution Styles

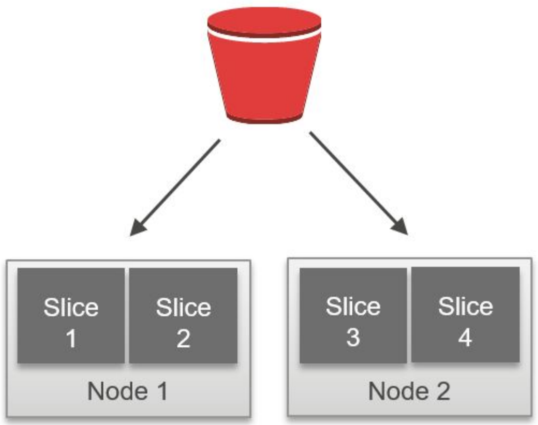
Distribution Key

Same key to same location
(Larger tables)



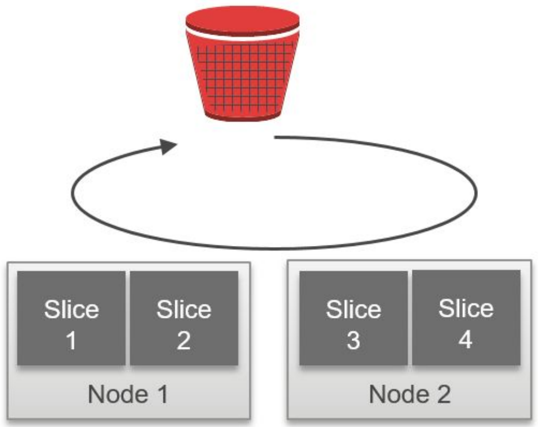
All

All data on every node
(Smaller tables)



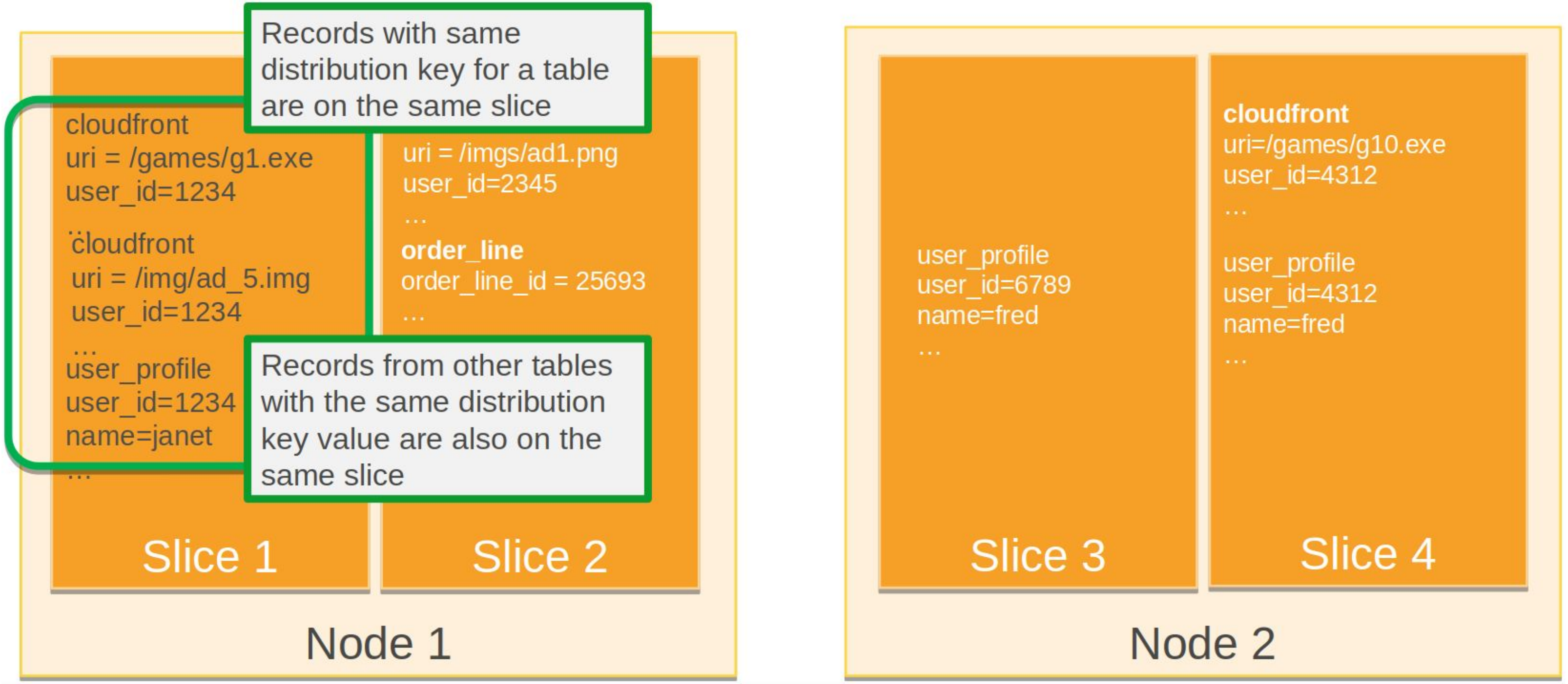
Even

Round robin distribution
(Tables not participating in JOINs or GROUP BYs)

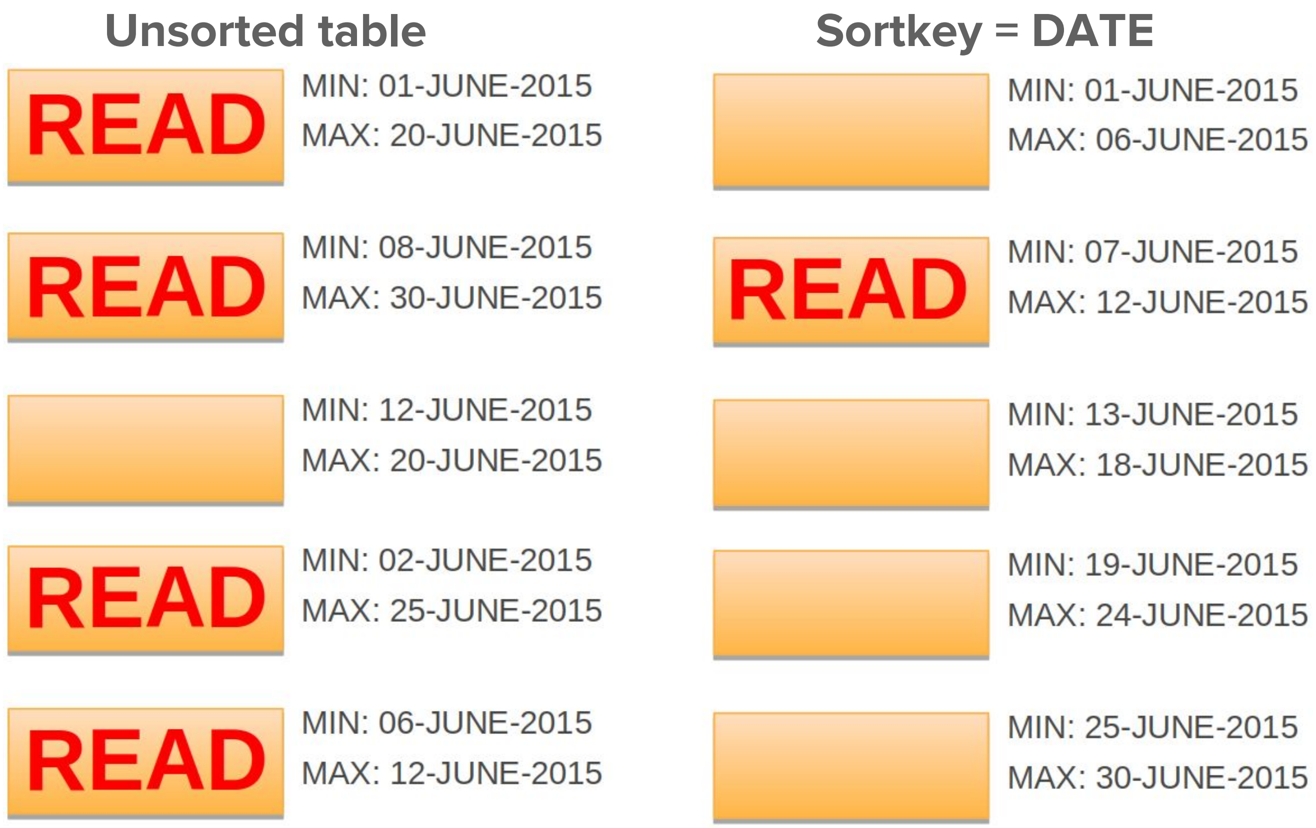


Distribution Key

Distribution key determines which data resides on which slices



Sort Key

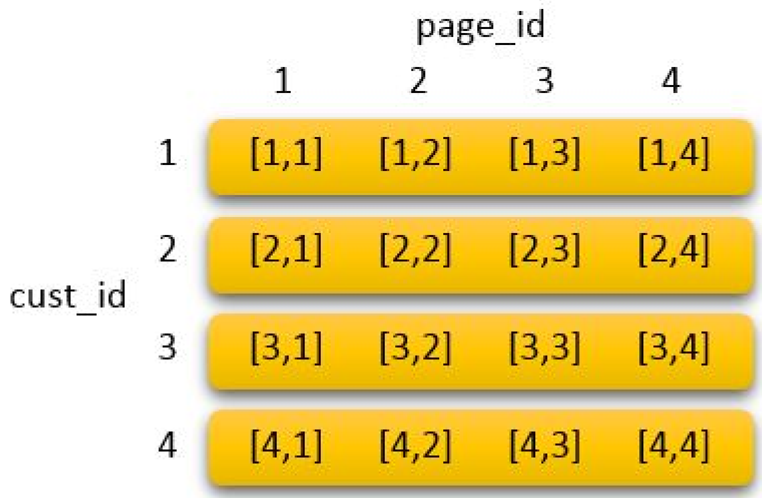


**SELECT COUNT(*)
FROM LOGS
WHERE DATE =
‘09-JUNE-2015’**

Sort Key - Multiple columns

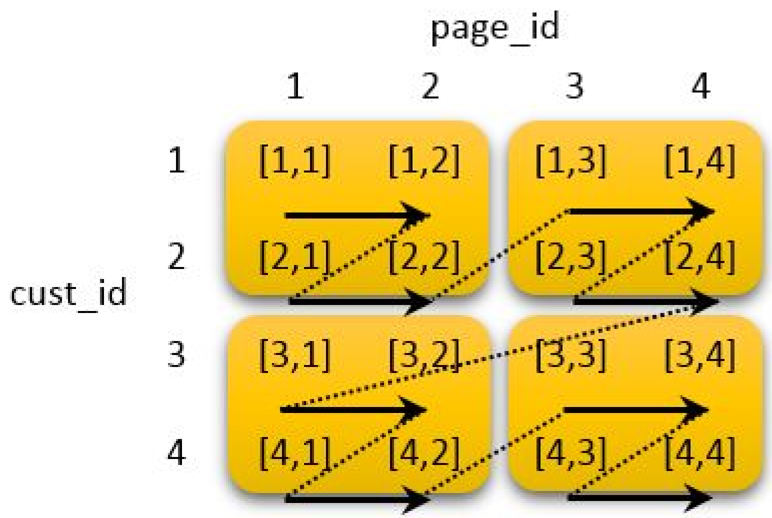
Compound Key

Sorted by cust_id, then by page_id
(Use when cust_id always in filter)



Interleaved Key

Both columns have equal weight:
(Use when either could be independently used in filter)



Interleaved keys are SLOW to vacuum

Chores

VACUUM - Reclaims space and resorts rows in either a specified table or all tables in the current database.

ANALYZE - Updates table statistics for use by the query planner.

Backups

Snapshots

- Automated snapshots occur every 8 hours or 5GB and retained for a configurable period
- Snapshots are stored in S3 and can be stored on a different region than the cluster
- Manual snapshots can be triggered
- Can restore full snapshot to new cluster or restore single table to existing cluster

UNLOAD command

- Query results can be exported to a file on S3

(Some) Interactive Clients

1 | JetBrains DataGrip

Pros

- Amazing editor
- Cross-platform (Java)

Cons

- Commercial
- No Redshift DDL support

2 | SQL Workbench/J

Pros

- Open Source
- Cross-platform (Java)

Cons

- Editor feels dated
- No Redshift DDL support

3 | Aginity Workbench

Pros

- Native Redshift support including DDL

Cons

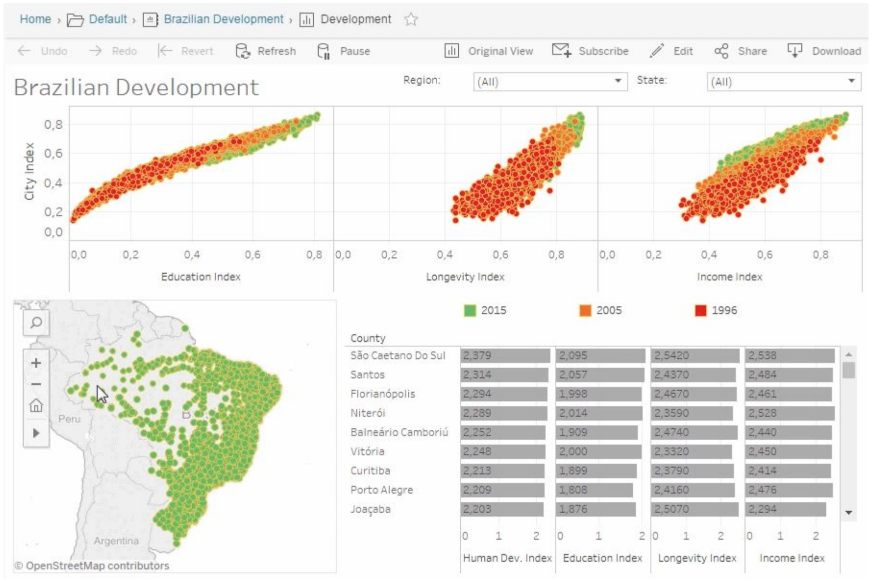
- Closed-source
- Windows-only

(Some) BI Tools with Native Support

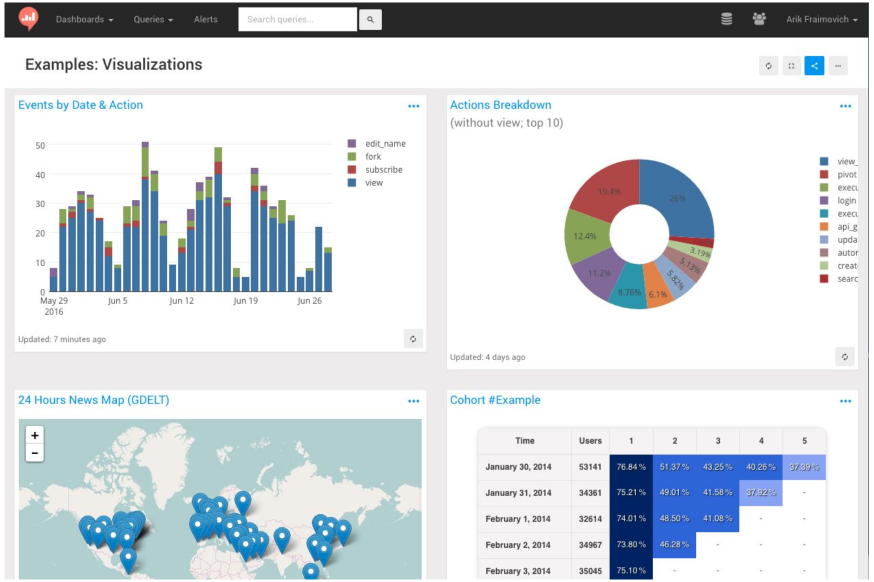
Tableau

re:dash

(Open Source)



County	Human Dev. Index	Education Index	Longevity Index	Income Index
São Caetano Do Sul	2,379	2,095	2,5420	2,538
Santos	2,314	2,057	2,4370	2,484
Florianópolis	2,294	1,998	2,4670	2,461
Niterói	2,289	2,014	2,3590	2,528
Balneário Camboriú	2,252	1,909	2,4740	2,440
Vitória	2,248	2,000	2,3320	2,450
Curitiba	2,213	1,899	2,3790	2,414
Porto Alegre	2,209	1,808	2,4160	2,476
Joaçaba	2,203	1,876	2,5070	2,294

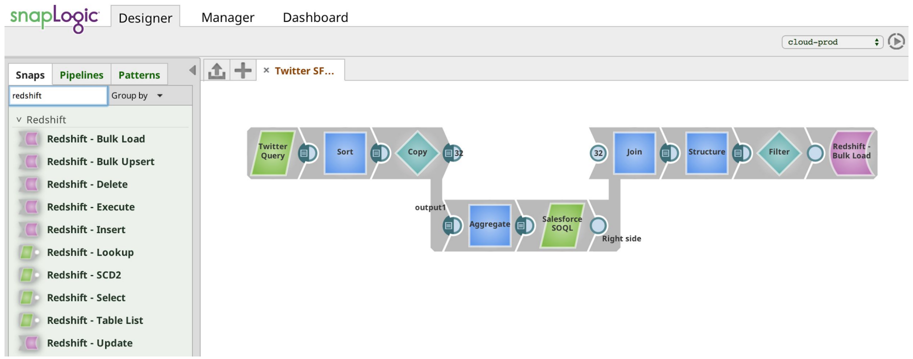


Time	Users	1	2	3	4	5
January 30, 2014	53141	76.84 %	51.37 %	43.25 %	40.26 %	37.39 %
January 31, 2014	34361	75.21 %	49.01 %	41.58 %	37.92 %	-
February 1, 2014	32614	74.01 %	48.50 %	41.08 %	-	-
February 2, 2014	34967	73.80 %	46.28 %	-	-	-
February 3, 2014	35045	75.10 %	-	-	-	-

TONS more @ https://aws.amazon.com/redshift/partners/

(Some) Data Integration Tools with Native Support

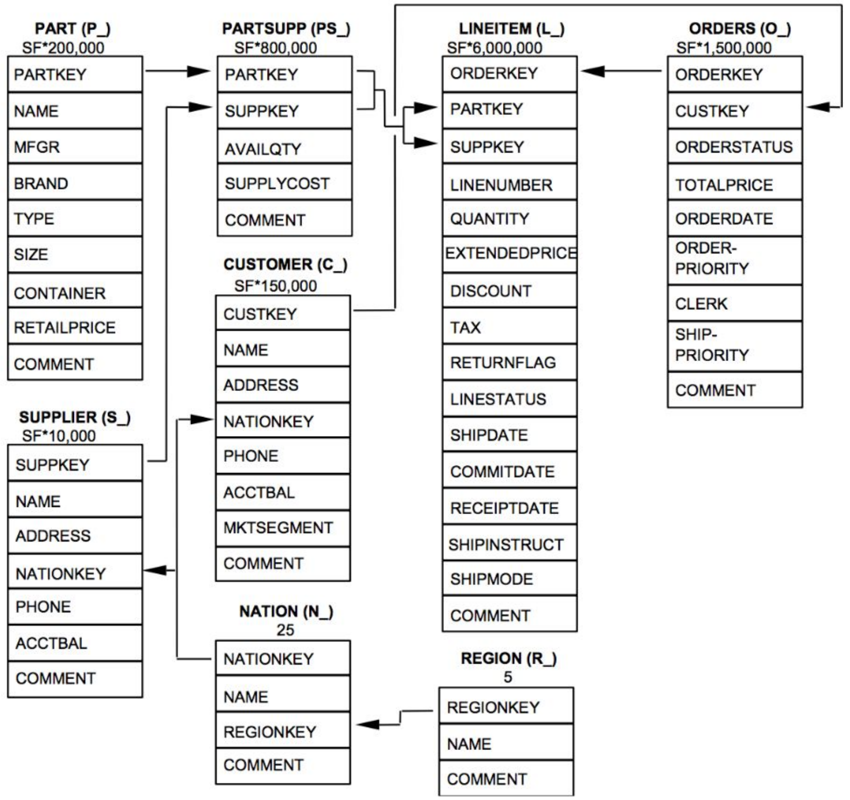
Informatica



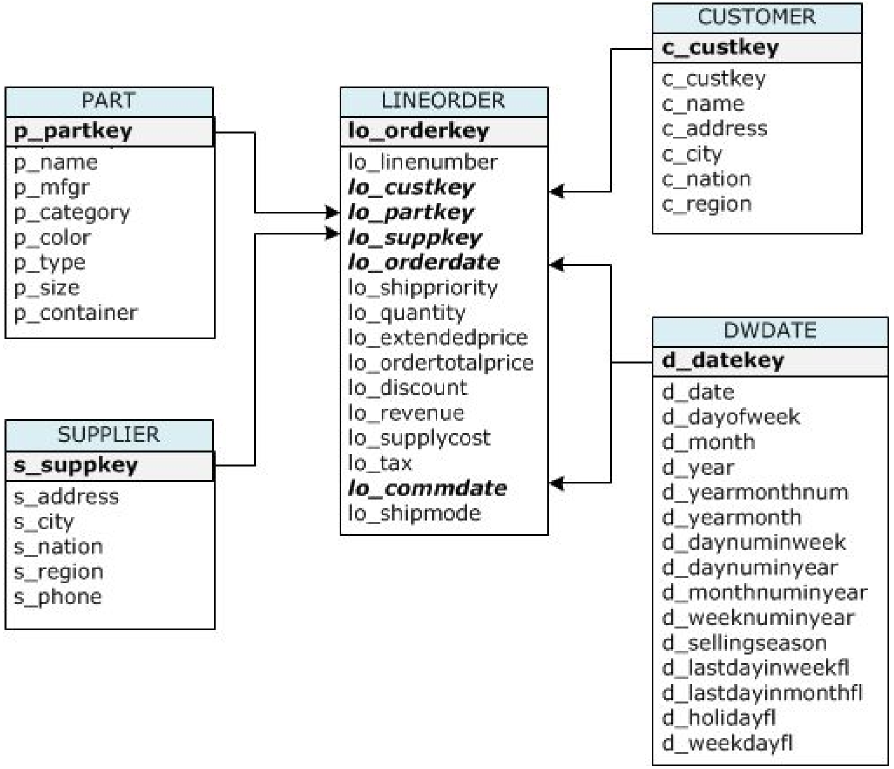
SnapLogic

TONS more @ https://aws.amazon.com/redshift/partners/

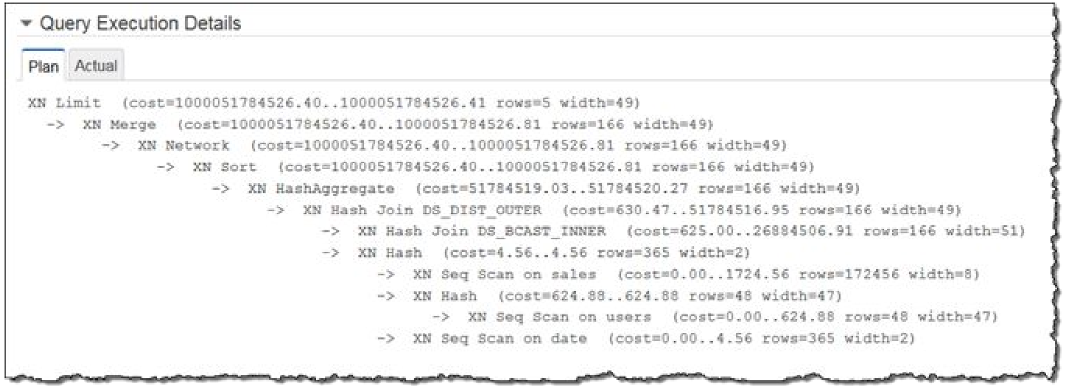
Normalized schema (TPC-H Benchmark)



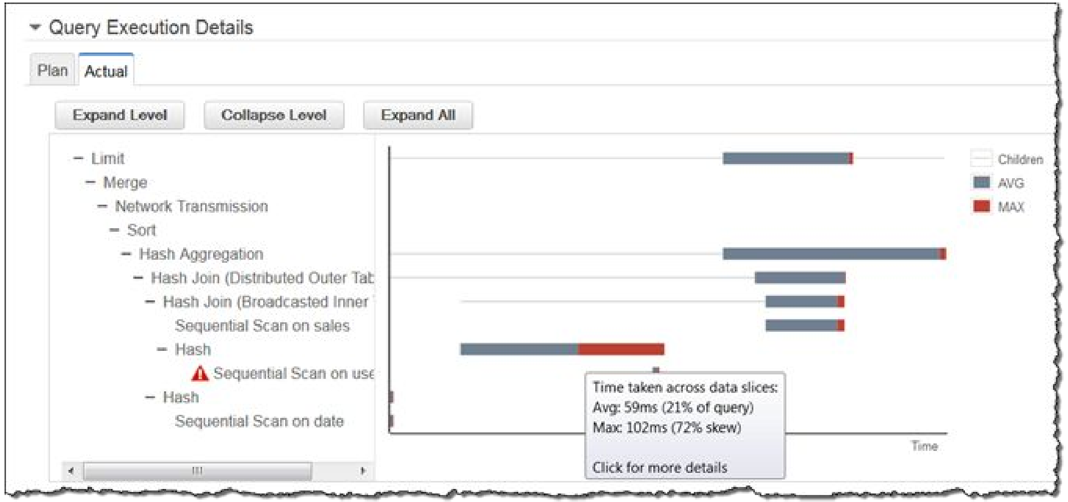
Star schema (Star Schema Benchmark)



Performance Analysis



```
XN Limit  (cost=1000051784526.40..1000051784526.41 rows=5 width=49)
  ->  XN Merge  (cost=1000051784526.40..1000051784526.81 rows=166 width=49)
        ->  XN Network  (cost=1000051784526.40..1000051784526.81 rows=166 width=49)
              ->  XN Sort  (cost=1000051784526.40..1000051784526.81 rows=166 width=49)
                    ->  XN HashAggregate  (cost=51784519.03..51784520.27 rows=166 width=49)
                          ->  XN Hash Join DS_DIST_OUTER  (cost=630.47..51784516.95 rows=166 width=49)
                                ->  XN Hash Join DS_BCAST_INNER  (cost=625.00..26884506.91 rows=166 width=51)
                                ->  XN Hash  (cost=4.56..4.56 rows=365 width=2)
                                      ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
                                      ->  XN Hash  (cost=624.88..624.88 rows=48 width=47)
                                            ->  XN Seq Scan on users  (cost=0.00..624.88 rows=48 width=47)
                                      ->  XN Seq Scan on date  (cost=0.00..4.56 rows=365 width=2)
```



Time taken across data slices:
Avg: 59ms (21% of query)
Max: 102ms (72% skew)

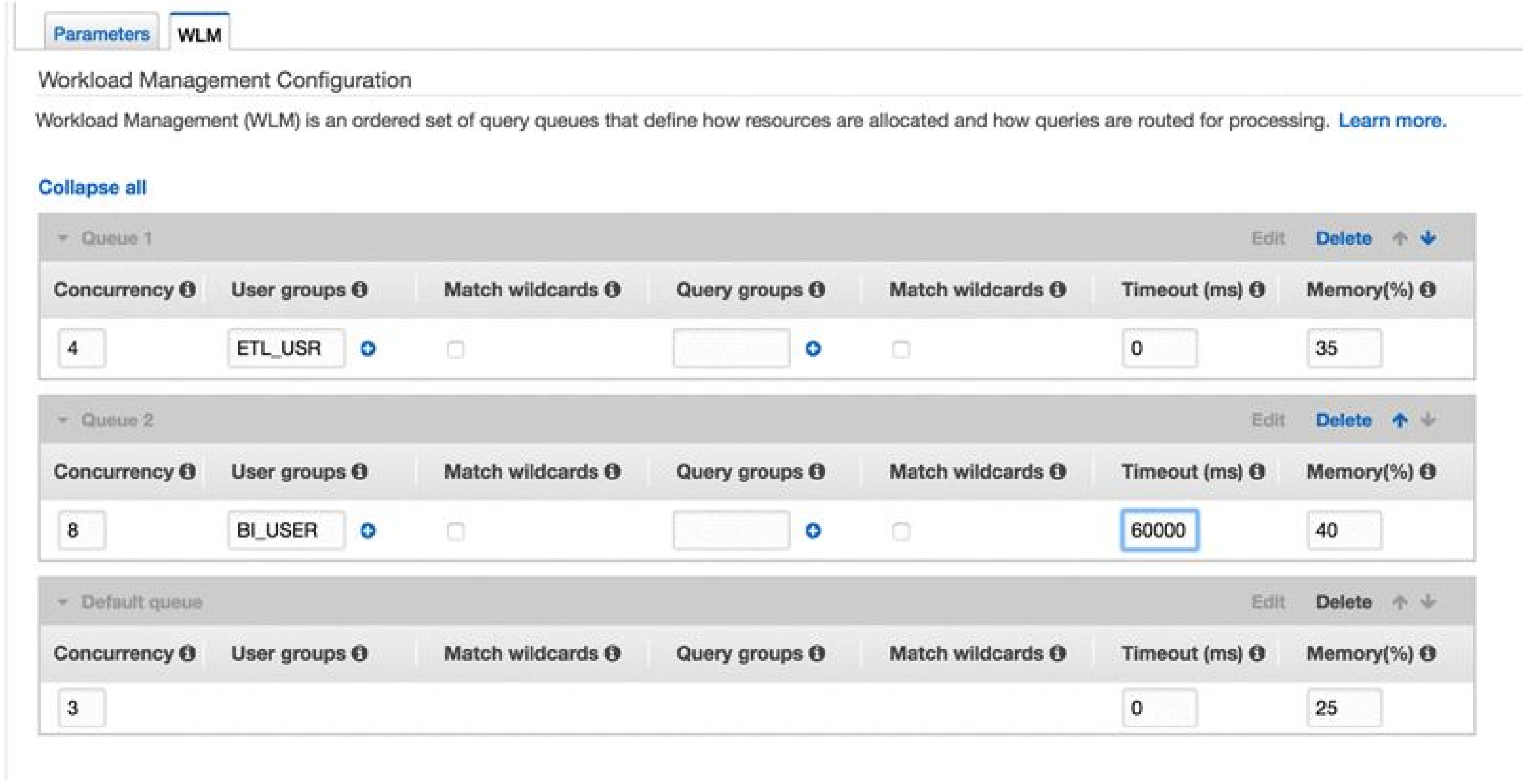
Click for more details

EXPLAIN command will display query execution plan

STL_EXPLAIN table stores execution plan for past queries

SVL_QUERY_REPORT view shows stats on queries already executed

Workload Management



Parameters | WLM

Workload Management Configuration

Workload Management (WLM) is an ordered set of query queues that define how resources are allocated and how queries are routed for processing. Learn more.

Collapse all

Queue 1 — Edit Delete

Concurrency	User groups	Match wildcards	Query groups	Match wildcards	Timeout (ms)	Memory(%)
4	ETL_USR	☐		☐	0	35

Queue 2 — Edit Delete

Concurrency	User groups	Match wildcards	Query groups	Match wildcards	Timeout (ms)	Memory(%)
8	BI_USER	☐		☐	60000	40

Default queue — Edit Delete

Concurrency	User groups	Match wildcards	Query groups	Match wildcards	Timeout (ms)	Memory(%)
3					0	25

Redshift SQL Gotchas

Relational constraints (primary/foreign keys) are informational for query planning but aren’t enforced.

Don’t use views - duplicate the data instead. Filters not propagated down.

No control statements (IF/ELSE)

No secondary indexes

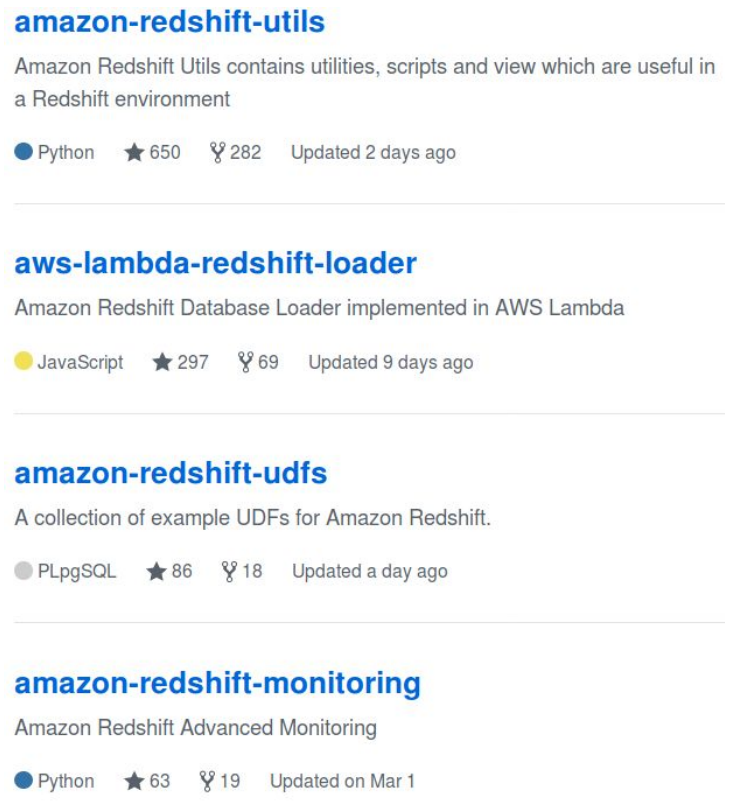
No stored procedures or triggers

User Defined Functions exist, but written in Python

Temp tables possible but slow to create and populate

Tools

https://github.com/awslabs



amazon-redshift-utils

Amazon Redshift Utils contains utilities, scripts and view which are useful in a Redshift environment

● Python ★ 650 282 Updated 2 days ago

aws-lambda-redshift-loader

Amazon Redshift Database Loader implemented in AWS Lambda

● JavaScript ★ 297 69 Updated 9 days ago

amazon-redshift-udfs

A collection of example UDFs for Amazon Redshift.

● PLpgSQL ★ 86 18 Updated a day ago

amazon-redshift-monitoring

Amazon Redshift Advanced Monitoring

● Python ★ 63 19 Updated on Mar 1

References

https://aws.amazon.com/redshift/developer-resources/

https://aws.amazon.com/blogs/aws/quickly-filter-data-in-amazon-redshift-using-interleaved-sorting/

http://www.cs.umb.edu/~poneil/StarSchemaB.PDF

https://www.slideshare.net/AmazonWebServices/amazon-redshift-optimizing-performance-20150721

https://www.quora.com/What-is-the-difference-between-redshift-and-RDS