TABLEAU
CONFERENCE
Amazon Redshift and Tableau: Shifting to High Gear

Adiascar Cisneros (he/him/his)
Partner Sales Consultant
Tableau Software

#data19
Takeaways

- Understand the elements that affect “performance”
- Understand how data gets from Redshift to Tableau
- Learn how to optimize Redshift for Tableau
- Learn how to optimize Tableau for Redshift
About Tableau and Redshift
With Tableau, you just hook it up to the Redshift server, connect, run a query, and publish it to the Server and you're literally done in an hour. It’s great—it feels like one product.

Abhishek Gupta
Senior Analyst, Box
Deploying Tableau on Amazon Redshift was an enterprise-wide transformation [...] resulting in operational efficiencies, more strategic programs and partnerships [...] , and ultimately, driving a data-enabled culture.

Jason Lokkesmoe
AVP, Big Data & Analytics Business Development, Pearson
Case Study: Pearson

Case Study: Pearson

Pearson’s architecture as of 2019

Where to Start?
Mapping the Challenge

Live Connection

Amazon Redshift

Extract

Desktop

Server

Nodes

Processes

Dashboard

Perceived Performance

- Infrastructure
- Security
- External Factors
Optimize Redshift and Tableau

1. Redshift Architecture
2. Distribution Styles
3. Table Design
4. Sort Keys
5. Compression
6. Encryption
7. WLM and Concurrency
8. VACUUM & ANALYZE
9. Cursors
10. Tableau Extracts
Redshift Architecture

Redshift Cluster

Compute Node 1

Node Slices

Compute Node n

Node Slices

Leader Node

JDBC

ODBC

Client Applications

Desktop

Server
Redshift Tips: Architecture

Use Dense Compute nodes (DC2)
  Efficient at running complex queries

Scale out
  Parallelize query execution

Consider Redshift Elastic Resize
  Add or remove nodes in minutes with little interruption
Tableau Tips: Architecture

Aggregate measures
Take advantage of Redshift’s compute power

Aggregate dates
Retrieve data at the analysis granularity, not capture granularity
Distribution Styles

**All**
Replicate to each node

**Key**
Same key on same node

**Even**
Round Robin
Redshift Tips: Distribution

Use AUTO distribution when possible
  Based on table size. Starts with ALL and can change to EVEN

Use KEY Distribution for large tables
  Avoid Table Data Skew

Use ALL Distribution for small tables
  Small tables are local for joins

Use EVEN Distribution as last resort
  When the table does not participate in joins or no choice is clear
Relational Database Disk Storage

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>101259797</td>
<td>SMITH</td>
<td>88</td>
<td>899 FIRST ST</td>
<td>JUNO</td>
<td>AL</td>
</tr>
<tr>
<td>892375862</td>
<td>CHIN</td>
<td>37</td>
<td>16137 MAIN ST</td>
<td>POMONA</td>
<td>CA</td>
</tr>
<tr>
<td>318370701</td>
<td>HANDU</td>
<td>12</td>
<td>42 JUNE ST</td>
<td>CHICAGO</td>
<td>IL</td>
</tr>
</tbody>
</table>
### Columnar Database Disk Storage

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Age</th>
<th>Address</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>101259797</td>
<td>SMITH</td>
<td>88</td>
<td>899 FIRST ST</td>
<td>JUNO</td>
<td>AL</td>
</tr>
<tr>
<td>892375862</td>
<td>CHIN</td>
<td>37</td>
<td>16137 MAIN ST</td>
<td>POMONA</td>
<td>CA</td>
</tr>
<tr>
<td>318370701</td>
<td>HANDU</td>
<td>12</td>
<td>42 JUNE ST</td>
<td>CHICAGO</td>
<td>IL</td>
</tr>
</tbody>
</table>

### Block 1

<table>
<thead>
<tr>
<th>SSN</th>
<th>892375862</th>
<th>318370701</th>
</tr>
</thead>
<tbody>
<tr>
<td>101259797</td>
<td>892375862</td>
<td>318370701</td>
</tr>
<tr>
<td>468248180</td>
<td>378568310</td>
<td>231346875</td>
</tr>
<tr>
<td>317346551</td>
<td>770033652</td>
<td>277332171</td>
</tr>
</tbody>
</table>

Table Design
Redshift Tips: Design

Keep your tables narrow

Few columns
Smallest possible column width
Tableau Tips: Design

Create aggregate tables

Avoid repeating operations for known queries of static data
Sort Keys

Sort keys determine the order to store data on disk

**Single Column**
Simple and fast

**Compound**
Most common
Best for known query patterns

**Interleaved**
Compromise for variable query patterns and large tables (> billion rows)
Single Column Sort Key

Region | (other columns) | Block
--- | --- | ---
North (N) |  | 
North (N) |  | 
North (N) |  | 
North (N) |  | 
South (S) |  | 
South (S) |  | 
South (S) |  | 
South (S) |  | 

Blocks

<table>
<thead>
<tr>
<th>N</th>
<th>S</th>
<th>E</th>
<th>W</th>
</tr>
</thead>
</table>

Sort Keys

1 2 3 4 5 6 7 8 9 10
### Compound Sort Key

<table>
<thead>
<tr>
<th>Region</th>
<th>Product</th>
<th>(other columns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>North (N)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>North (N)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>North (N)</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>North (N)</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>East (E)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>East (E)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>East (E)</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>East (E)</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

#### Blocks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[N,A]</td>
<td>[S,A]</td>
<td>[E,B]</td>
<td>[W,A]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[N,A]</td>
<td>[S,B]</td>
<td>[E,B]</td>
<td>[W,C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[N,C]</td>
<td>[S,B]</td>
<td>[E,C]</td>
<td>[W,D]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[N,D]</td>
<td>[S,D]</td>
<td>[E,D]</td>
<td>[W,D]</td>
</tr>
</tbody>
</table>

---

1. Sort Keys
# Interleaved Sort Key

<table>
<thead>
<tr>
<th>Region</th>
<th>Product</th>
<th>(other columns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>North (N)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>North (N)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>South (S)</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Product</th>
<th>(other columns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>East (E)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>East (E)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>West (W)</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>West (W)</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

### Blocks

- **North (N)**: [N,A] [N,B] [N,C] [N,D]
- **South (S)**: [S,A] [S,B] [S,C] [S,D]
- **East (E)**: [E,A] [E,B] [E,C] [E,D]
- **West (W)**: [W,A] [W,B] [W,C] [W,D]

### Sort Keys

1. North (N) A
2. North (N) B
3. South (S) A
4. South (S) B
5. East (E) A
6. East (E) B
7. West (W) A
8. West (W) B
9. East (E) C
10. East (E) D
Redshift Tips: Sort Keys

Use Compound for dashboards following Tableau best practices

  Ideal for diving into data details

Use Compound on timestamp if using recent data or data range queries

  Allows skipping blocks outside the range

Put Sort Keys in columns used as Quick Filters

  They require a query to fill the filter values
Redshift Tips: Keys for Joining Tables

Use join column as Sort Key and Distribution Key for frequently joined tables

Allows sort merge join instead of slower hash join
Tableau Tips: Sort Keys

Join tables using sort keys

- Tableau can then use join culling
- Non-DB admins can turn on “Assume Referential Integrity” to get similar effect

Use sort keys to support filters

- Reduce scanning to build the filter options list

Joining columns should be defined as NOT NULL

- Null value presence requires more complex queries

Use sort keys for Level of Detail Expressions (LOD)

- LOD can generate performance-degrading cross joins
Compression

Column level

Compressing data types separately yields better results

Applied during table creation or COPY operation

For existing tables, recreate it with new compression settings

Increases performance

More data transferred with each read

Reduces cost

Data fits in a smaller cluster
Redshift Tips: Compression

Don’t compress the single column sort key

Don’t compress the first column in a compound sort key
Redshift Tips: Compression

Use COPY when loading data to apply compression automatically
  COPY automatically chooses best compression encoding per column

Disable COMPUPDATE for staging and temp tables
  Avoid unnecessary overhead during ETL process
Encryption

Encryption is expensive

20% performance impact is typical

Key Management

AWS KMS supported starting with Tableau Server 2019.3
Redshift (and Tableau) Tips: Encryption

Test performance with and without encryption

Determine the impact in your environment as a reference
Workload Management (WLM)

WLM allows prioritization of query queues
   Fast-running queries won't get stuck behind long-running queries

Up to 8 user queues
   Compartmentalize fallout of poorly designed queries

Up to 50 concurrent queries
   Expand as needed

Auto or Manual WLM
   The machines can do it
Demand versus Capacity

Queue Length by WLM queues
Demand versus Capacity

Allocated Capacity

Wasted Capacity

Queue Length by WLM queues

Demand
Concurrency Scaling

Queue Length by WLM queues ➤ Active Concurrency Scaling Clusters

WLM and Concurrency
Redshift Tips: WLM and Concurrency

Enable Auto WLM

For Manual WLM, configure 15 or less concurrent slots across all queues when concurrency scaling is disabled

Enable Concurrency Scaling

To handle an increase in concurrent read queries

Enable Short Query Acceleration

Short queries aren't forced to wait in queues behind longer queries

Create Query Monitoring Rules (QMR)

To track and handle poorly written queries
Tableau Tips: WLM and Concurrency

Begin with summaries
Start with top level data, drill only when necessary

Simple dashboards
Keep number of queries low

Start with smallest set of data on filters
Avoid starting with “All”

Provide an “Apply” filter button
Otherwise, Multiple Selection filters run a query every time an item is clicked
VACUUM and ANALYZE

VACUUM recovers space and restores sort order
Just say “no” to performance degradation

ANALYZE updates the table statistics
Get more accurate query plans, queries are rewritten for efficiency

VACUUM DELETE is run automatically during off-peak times
It’s like brushing your teeth, do it often

ANALYZE is run automatically on a regular schedule
Runs hourly
Redshift Tips: VACUUM and ANALYZE

Run VACUUM after significant additions, updates, or deletes
Clean up after yourself

Set STATUPDATE ON to ANALYZE automatically after data loads
Increases accuracy of query optimizer

COPY sequential files in order, to avoid the need to VACUUM
If you’re going to do it, do it right the first time
Cursors

Retrieve more data than you have memory

Bite more than you can chew

Performance impact

All data funneled through lead node

Cursor limits

Dependent on instance type
Redshift Tips: Cursors

Increase concurrent cursors

In the Redshift database
Tableau Tips: Cursors

Increase concurrent cursors
   Per data source or per workbook

Aggregate measures in workbooks
   Transfer less data
Tableau Tips: Extracts

Extract small data portions
  Only the “slices” that are relevant

Aggregate extracts
  Remove fields or use higher level of detail

Avoid scheduling extract refreshes in parallel
  Avoid hitting the open cursor size limit
Tableau Tips: Extracts

Check the size of the extract in advance

- Results are materialized on the Leader node of your cluster

Consider other users

- Cursor errors affect all users

Consider large nodes

- More cursor space
Summary

- Consider the entire data journey
  Start with Redshift and work your way to your dashboards

- Optimize tables for most common queries
  Reduce the volume of data scanned

- Simplify your dashboards
  We don’t need all the data all the time
Optimizing your Amazon Redshift and Tableau Software Deployment for Better Performance
Best Practices for Dashboard Performance
Nov 14 | 4:00 PM–5:01 PM

Tuning Tableau Server: Performance Best Practices
Nov 14 | 2:15 PM–3:16 PM
Nov 15 | 12:15 PM–1:16 PM
Additional References

Whitepapers

1. Tableau at the Speed of Amazon EC2

Customer Case Studies

2. Box
3. Pearson

AWS Blogs

4. Best practices for Amazon Redshift
5. Best practices for Amazon Redshift Spectrum
6. Best practices for designing ETL

Additional References

Sort Keys and Distribution References
1. Tutorial: Tuning Table Design
2. Advanced Table Design Playbook
3. Redshift Advisor

Workload Management (WLM)
4. Concurrency Scaling
5. Short Query Acceleration (SQA)
6. Create Query Monitoring Rules (QMR)

Additional References

Compression
1. Choosing a Column Compression Type
2. Loading Tables with Automatic Compression
3. Run Analyze Compression
4. Redshift Advisor

Encryption
5. Amazon Redshift Database Encryption

Cursors
6. Cursor constraints
7. Working with Amazon Redshift Concurrent Cursor Limit

6. https://docs.aws.amazon.com/redshift/latest/dg/declare.html#declare-constraints
7. https://community.tableau.com/docs/DOC-6323
Additional References

VACUUM and ANALYZE

1. Analyzing Tables
2. Vacuuming Tables
3. Automatic ANALYZE
4. Automatic VACUUM DELETE

1. https://docs.aws.amazon.com/redshift/latest/dg/t_Analyzing_tables.html
Please complete the session survey in the mobile app

View ‘My Evaluations’ in the menu or find your session under ‘Schedule’
Thank You

acisneros@tableau.com