Meet Redshift, a columnar database

INTRODUCTION TO REDSHIFT

Jason Myers
Principal Architect

datacamp

Redshift overview

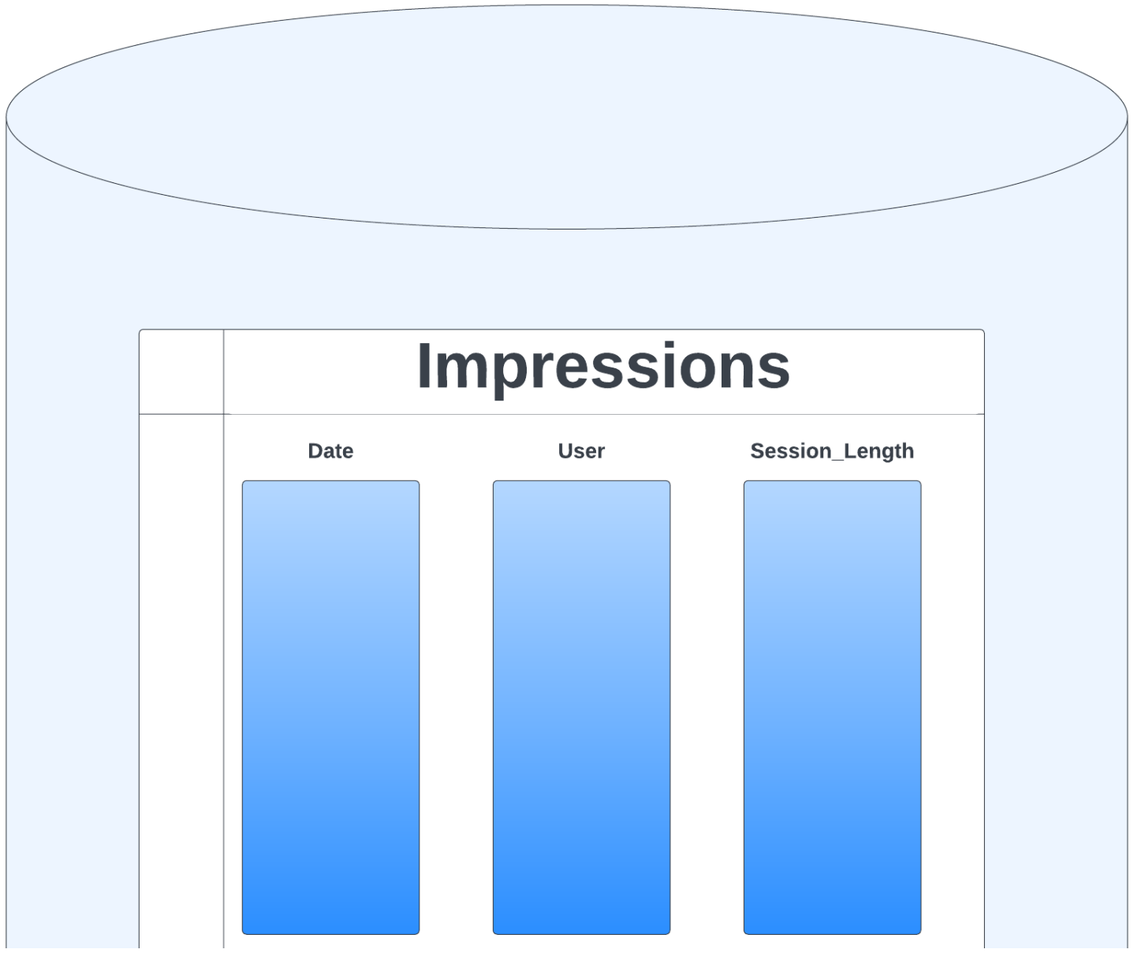
- Distributed
- Columnar database
- Uses PostgreSQL 9 syntax with some enhancements
- Serverless and provisioned clusters

amazon
REDSHIFT

Columnar vs row based databases

Row-based

Columnar



Redshift's position in the AWS ecosystem

- Primary SQL data warehouse
- Can be the center of our AWS environment for data lakehouse capabilities
- Supports external schemas in
  - AWS RDS for PostreSQL
  - AWS Aurora PostgreSQL-compatible edition
  - Amazon EMR for Hive support
  - Amazon Kinesis for streaming data support

Redshift's position in the AWS ecosystem

- Federated queries with other AWS RDS databases and more
- Integrates with Amazon SageMaker via Redshift ML to empower data scientists.

Competitors

- Snowflake, BigQuery, Databricks Lakehouse, Azure Synapse Analytics

Advantages

- AWS Integration: no copy data sharing with other services
- Share-nothing architecture
- Optimized repeat query performance
- Reserved instance (RI) pricing

Disadvantages

- AWS only
- Unoptimized query performance
- Flexibility to support external files costs extra

Viewing databases and schemas

- Use SVV_REDSHIFT_SCHEMAS (internal) or SVV_ALL_SCHEMAS (internal and external)

```
-- View the database and schema names with the schema type
SELECT database_name,
       schema_name,
       schema_type
  -- From the internal schemas view
  FROM SVV_REDSHIFT_SCHEMAS;
```

```
database_name        | schema_name         | schema_type
=====================|=====================| ===========
datacamp_course_prod | information_schema  | local
datacamp_course_prod | pg_catalog          | local
datacamp_course_prod | public              | local
dev                  | information_schema  | local
dev                  | pg_catalog          | local
dev                  | public              | local
```

Viewing tables in a schema

- Use SVV_REDSHIFT_TABLES (internal) or SVV_ALL_TABLES (internal and external)

```
-- View the table name
SELECT table_name
  -- Using a view with both internal and external tables
  FROM SVV_ALL_TABLES
 -- In the external spectrumdb schema
 WHERE schema_name = 'spectrumdb';
```

```
table_name
======================
ecommerce_sales
global_power_plant_db
coffee_county_weather
idaho_site_id
idaho_samples
```

Let's practice!

INTRODUCTION TO REDSHIFT

Learning about Redshift data types and features

INTRODUCTION TO REDSHIFT

Jason Myers
Principal Architect

datacamp

Basic data types

Numeric types

- SMALLINT, INTEGER, BIGINT
- DECIMAL / NUMERIC
- DOUBLE PRECISION
- REAL

Character types

- CHAR
- VARCHAR

Datetime types

- DATE
- TIME, TIMETZ
- TIMESTAMP, TIMESTAMPTZ

Boolean type

- BOOLEAN

Special data types

SUPER type

- Semistructured data
  - Arrays
  - Tuples
  - Nested structures (e.g. JSON)
- Uses PartiQL to handle them
- 16MB max size

VARBYTE type

- binary data (BLOBs)
- Images and Videos

Unsupported PostgreSQL types:

PostgreSQL Type	Redshift Types
DATETIME	TIMESTAMP, TIMESTAMPTZ
SERIAL	INTEGER, BIGINT
UUID	VARCHAR
JSON	SUPER, VARCHAR
ARRAY	SUPER, VARCHAR
BIT	BOOLEAN, SMALLINT, VARCHAR

Viewing columns and their data types

```
-- View the column details
SELECT column_name,
       data_type,
       character_maximum_length AS character_max_len,
       numeric_precision,
       numeric_scale
  -- Using a view with both internal and external schema's columns
  FROM SVV_ALL_COLUMNS
 -- Only in the spectrumdb schema
 WHERE schema_name = 'spectrumdb'
   -- For the ecommerce_sales table
   AND table_name = 'ecommerce_sales';
```

Viewing columns and their data types (cont)

```
column_name                 | data_type         | character_max_len | numeric_precision | numeric_scale
============================|===================|===================|===================|==============
year_qtr                    | character varying | 100               | null              | null
total_sales                 | integer           | null              | 32                | 0
ecom_sales                  | integer           | null              | 32                | 0
percent_ecom                | real              | null              | null              | null
percent_change_quarter_total | real              | null              | null              | null
percent_change_quarter_ecom  | real              | null              | null              | null
percent_change_year_total    | real              | null              | null              | null
percent_change_year_ecom     | real              | null              | null              | null
```

Data type "compatibility"

- Implicit conversion: assignments, comparisons
- Make sure to double check outcomes!

From Type	To Types
CHAR	VARCHAR
DATE	CHAR, VARCHAR, TIMESTAMP, TIMESTAMPTZ
TIMESTAMP	CHAR, DATE, VARCHAR, TIMESTAMPTZ
BIGINT	BOOLEAN, CHAR, DECIMAL, DOUBLE PRECISION, INTEGER, REAL, SMALLINT, VARCHAR
DECIMAL	BIGINT, CHAR, DOUBLE PRECISION, INTEGER, REAL, SMALLINT, VARCHAR

[1] https://docs.aws.amazon.com/redshift/latest/dg/c_Supported_data_types.html#r_Type_conversion

Explicitly casting data types

- CAST

```
-- Casting a decimal to a integer and aliasing it
SELECT CAST(2.00 AS INTEGER) AS our_int;
```

```
our_int
========
2
(1 row)
```

TO functions

- TO_CHAR, TO_DATE, TO_NUMBER

```
-- Convert the string to a date
SELECT CAST('14-01-2024 02:36:48' AS DATE) AS out_date;
```

```
date/time field value out of range: "14-01-2024 02:36:48"
HINT:  Perhaps you need a different "datestyle" setting.
```

```
-- Parse the string to a date
SELECT TO_DATE('14-01-2024 02:36:48', 'DD-MM-YYYY') AS our_date;
```

```
our_date
========
2024-01-14
(1 row)
```

TO functions (cont)

```
-- Get the name of the month in a date
SELECT TO_CHAR(date '2024-01-14', 'MONTH') AS month_name;
```

```
month_name
========
JANUARY
(1 row)
```

- Datetime and Numeric format strings

[1] https://docs.aws.amazon.com/redshift/latest/dg/r_FORMAT_strings.html

Let's practice!

INTRODUCTION TO REDSHIFT

Inside the Redshift warehouse

INTRODUCTION TO REDSHIFT

Jason Myers
Principal Architect

datacamp

Agenda

- Redshift cluster internals
  - Node types
  - Leader specific functions
  - Redshift cluster storage
- Predicates
- Redshift spectrum
  - Database components
  - External tables

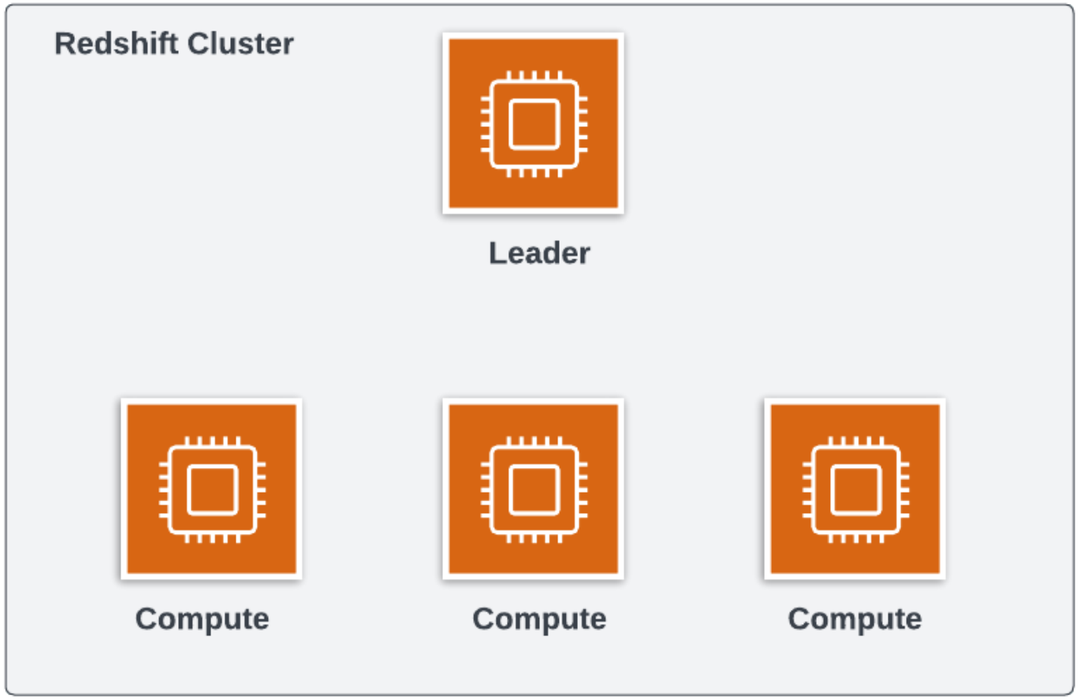
Redshift cluster architecture

Leader Node

- Provides connections
- Builds and distributes query execution plans
- Can execute entire queries
- Has exclusive functions

Compute Node

- Provides data storage
- Executes code from the leader node on locally stored data



Leader specific functions

- Only run on the leader

```
-- Selecting the substr, starting at
-- position 11 of 'chocolate chip'
SELECT SUBSTR('chocolate chip', 11);
```

```
chip
```

- Default to using `SUBSTRING`
- List of leader specific functions in **AWS Redshift documentation**

- `SUBSTR` errors on table columns

```
-- Selecting the substr from position 1
-- of the column named field on table
SELECT SUBSTR(field, 1) FROM table;
```

```
ERROR: SUBSTR() function is not
supported (Hint: use SUBSTRING
instead)
```

Looking at data across nodes

```
SELECT host,
       -- Calculate the percentage of used space
       -- using the used minus tossed or ready to be reclaimed
       -- divided by the capacity
       (used - tossed) / capacity * 100 as percent_used
  FROM STV_PARTITIONS;
```

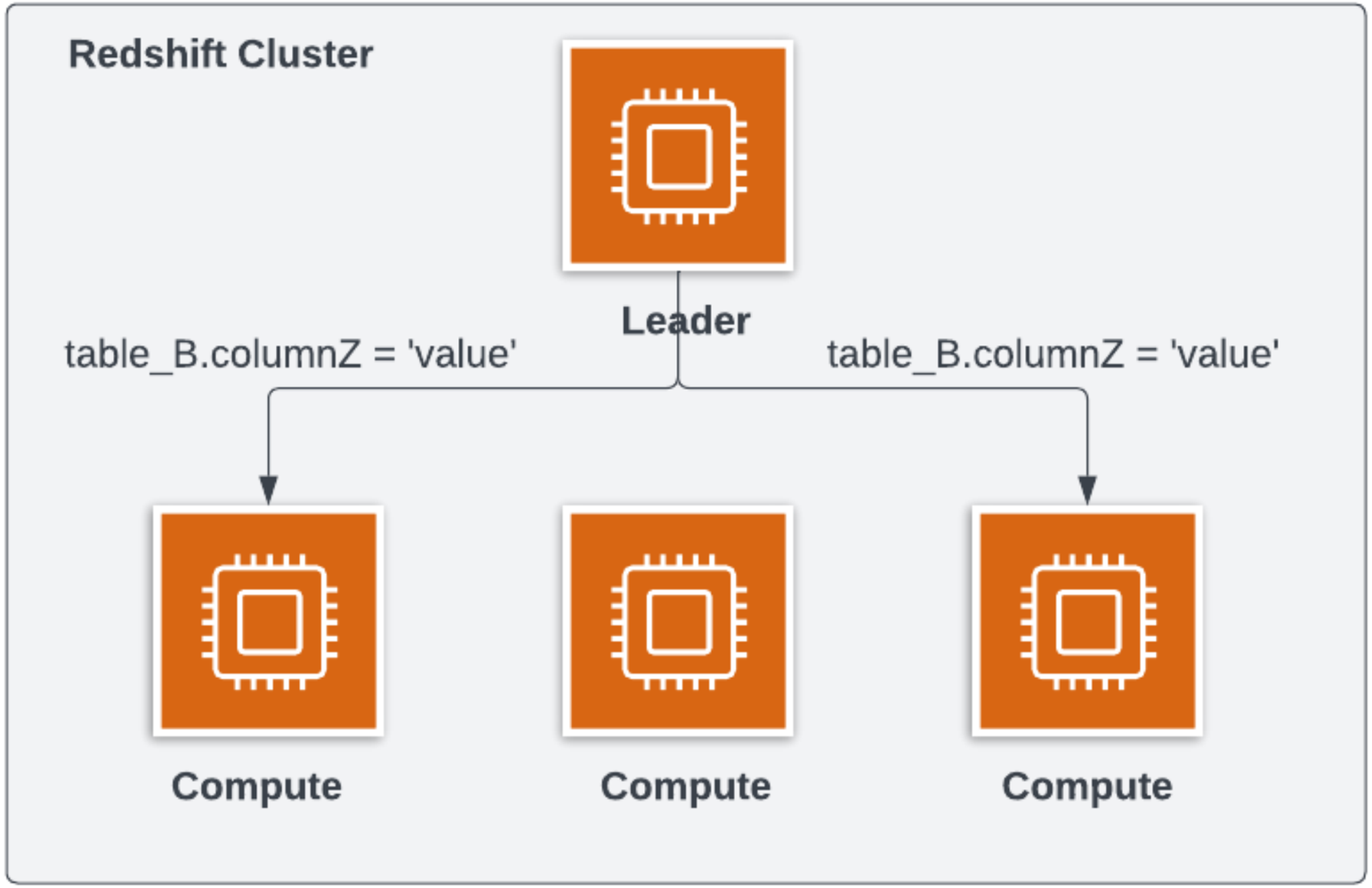
```
 host |  percent_used
======+==============
  0   |  24.9
  1   |  24.8
```

Predicates

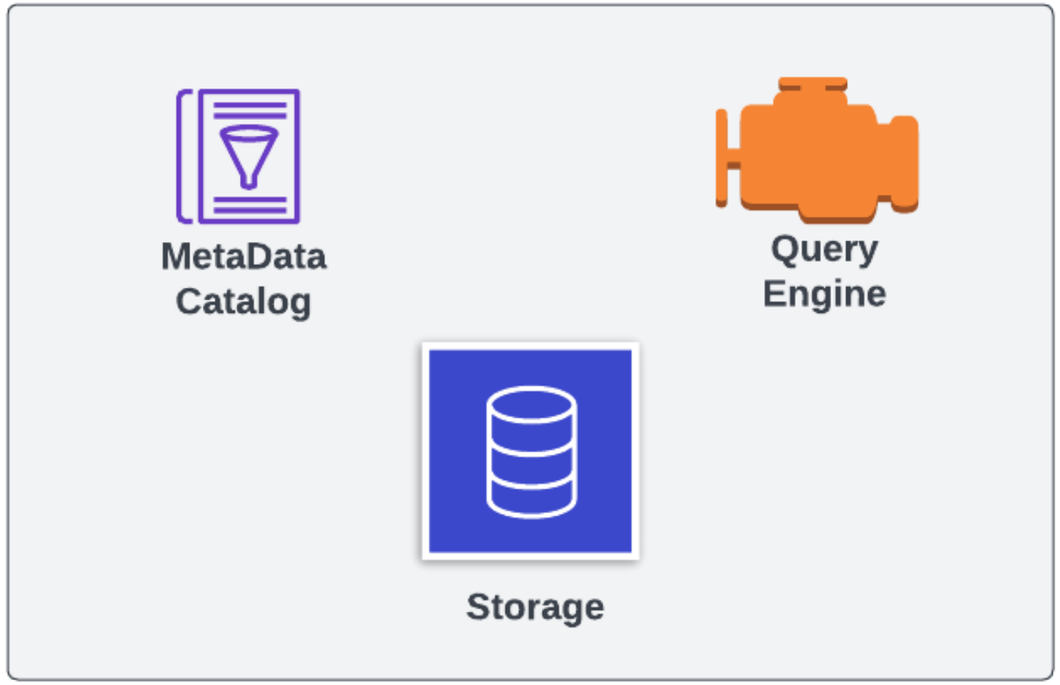
```
SELECT table_A.columnX,
       table_B.columnY,
  FROM table_A
       INNER JOIN table_B
          -- predicate
       ON table_B.foreign_key = table_A.primary_key
       -- predicate
 WHERE table_B.columnZ = 'value';
```

- Typically boolean expressions and found in WHERE, HAVING or ON SQL clauses

Predicate push-down



Typical internal database components



MetaData Catalog

- Holds schema info (columns, keys, etc)
- References a storage location

Query Engine

- Plans and executes queries
- Provides connections

Storage

- Hold table data
- Supports multiple file and table formats

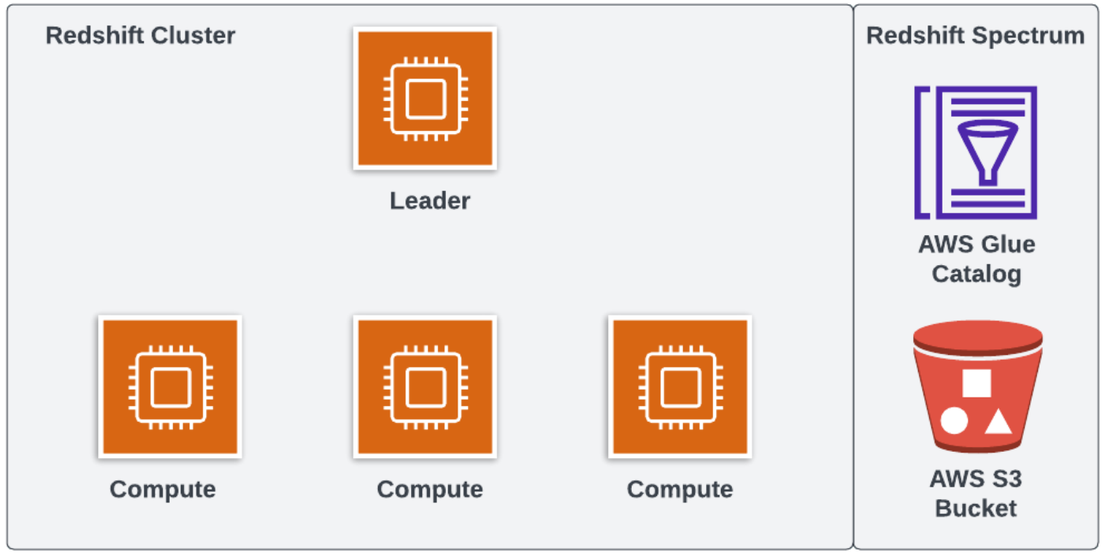
Redshift spectrum architecture

AWS Glue Data Catalog

- Stores information about "external" tables

AWS S3 Bucket

- Stores the files that represent the table
- Supports CSVs, JSON, Text, Parquet, and many more file types



Let's practice!

INTRODUCTION TO REDSHIFT