



2. Redshift  
Architecture  
and Key  
Concepts



1. Redshift  
Overview



3. Redshift  
Administration



4. Redshift  
Development



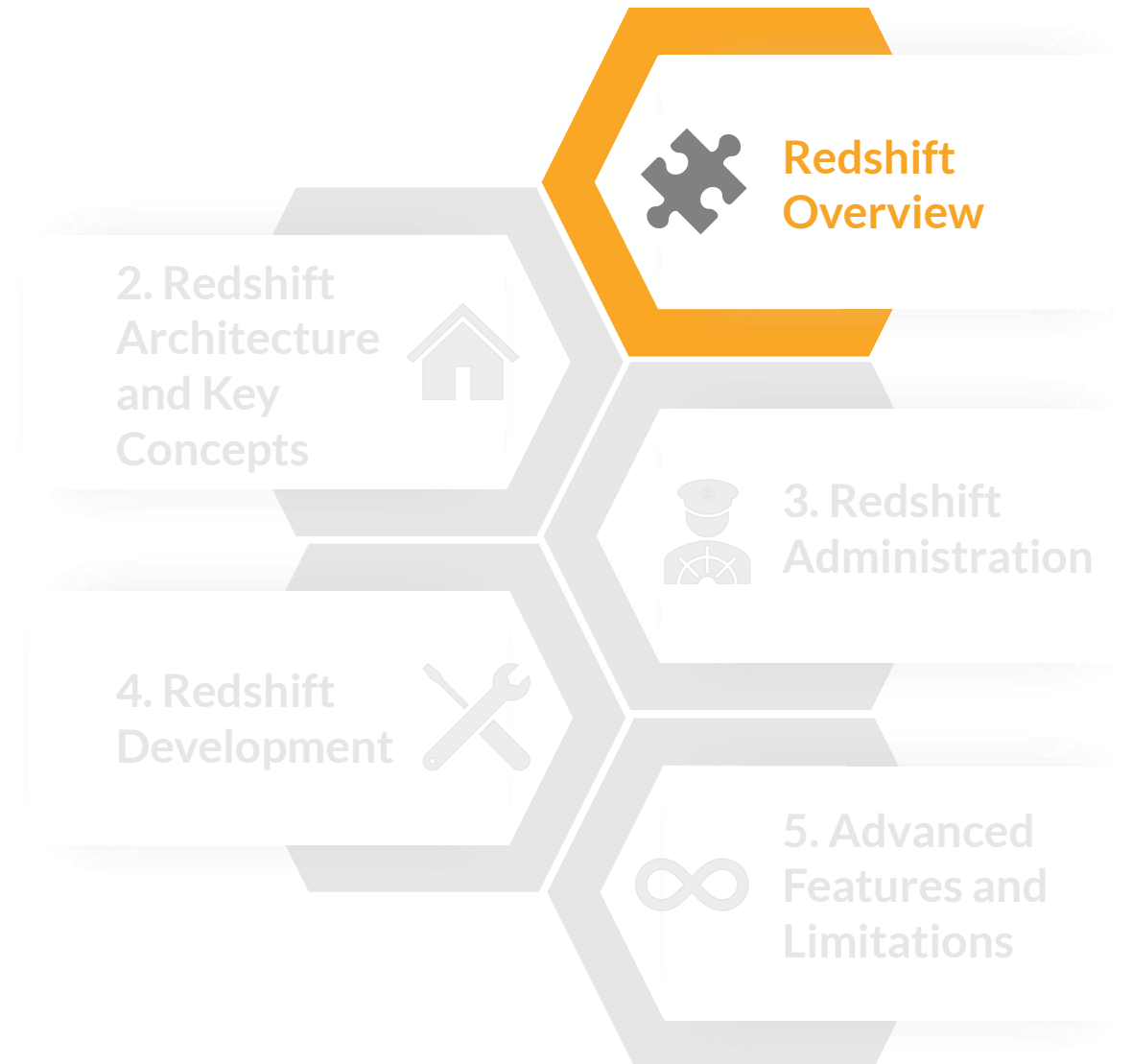
5. Advanced  
Features and  
Limitations

# Session 1: Redshift Overview

Upon completion of this session, you will be able to answer the following questions:

- What are the limitations of traditional data warehouses?
- Why industries choose Redshift as their data warehousing solution?
- What are some of the use-cases of Amazon Redshift?

We will also look at a case study of an industry using Amazon Redshift at the center of their data and analytics pipeline



# ABOUT DATA – QUICK RECAP

Data helps identify and drive insights, which helps in decision-making

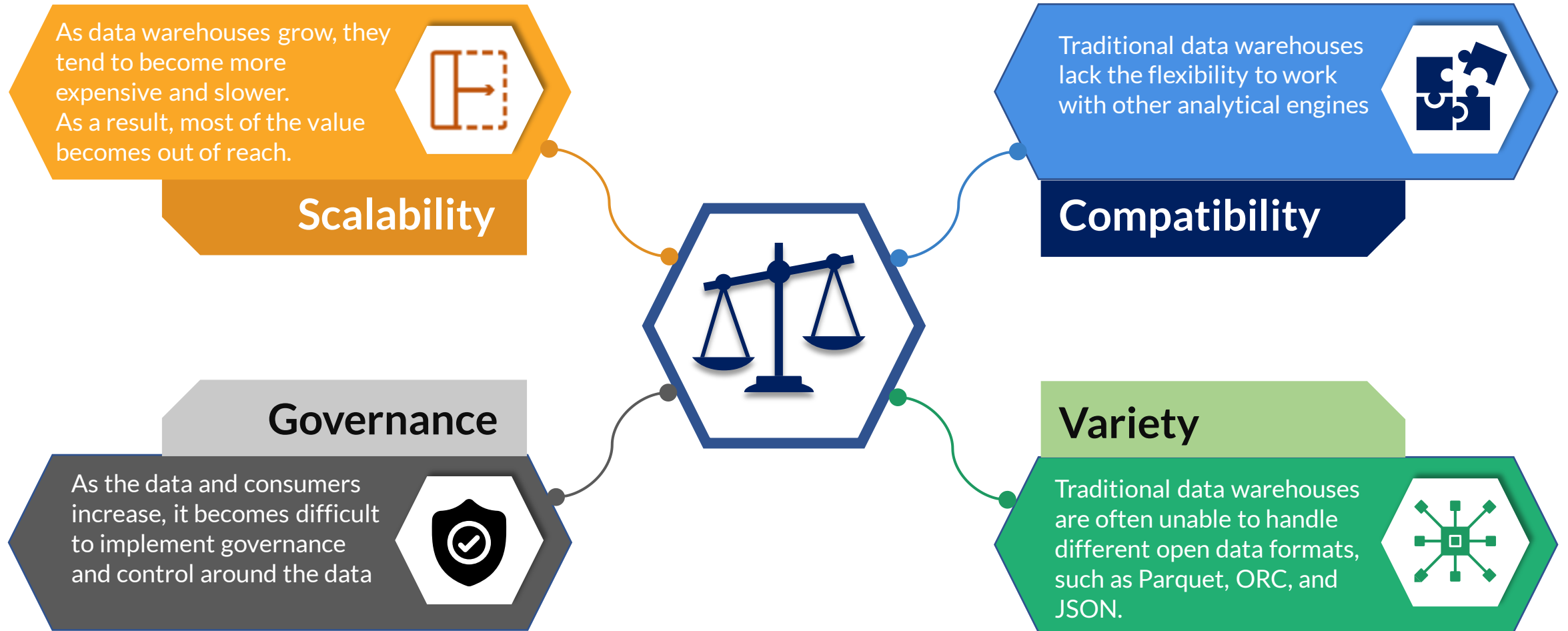
01 What is data?

02 Why do we need to store data?

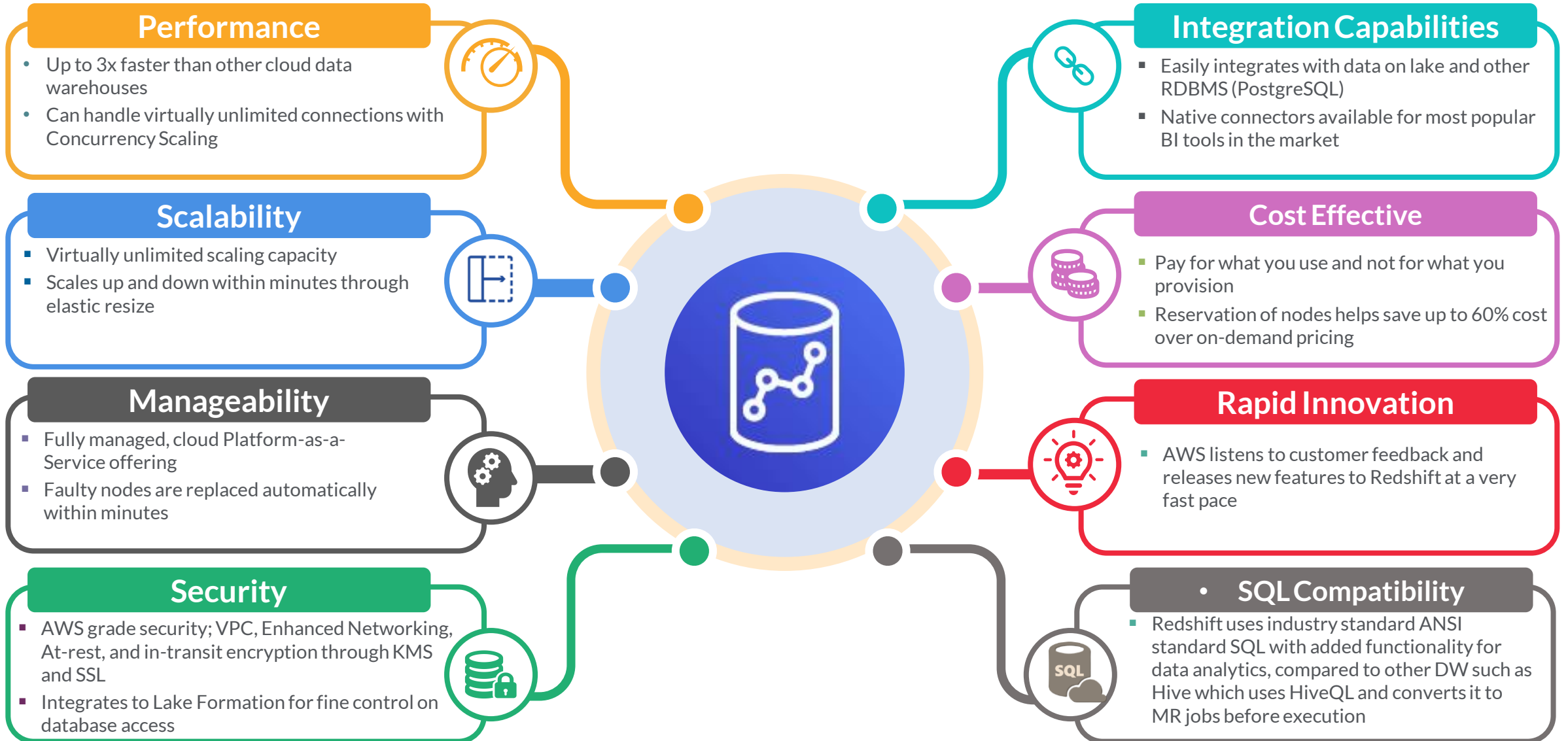
03 Where do we store data?













# LIMITATIONS OF TRADITIONAL DATA WAREHOUSES



# Why Redshift?



# INDUSTRIAL USE-CASES FOR AMAZON REDSHIFT

Industry	Use Cases	Example
Healthcare	<ul style="list-style-type: none"><li>Analyze clinical records to improve patient outcomes and predict diseases for preventive programs</li></ul>	 
Financial Services	<ul style="list-style-type: none"><li>Analyse trading and market data, risk analyses, fraud detection</li></ul>	 
Food and Manufacturing	<ul style="list-style-type: none"><li>Create personalised experiences and offers for customers</li></ul>	 
Gaming	<ul style="list-style-type: none"><li>Aggregate data from games and players and analyse in-game behaviour</li></ul>	
Telecommunications & Media	<ul style="list-style-type: none"><li>Store, process and analyse call data records for consumer billing</li><li>Analyse consumer behaviour for personalized recommendations</li></ul>	 
Advertising	<ul style="list-style-type: none"><li>Analyse clickstream and ad impression logs to improve ad targeting</li></ul>	

# CASE STUDY: Nasdaq

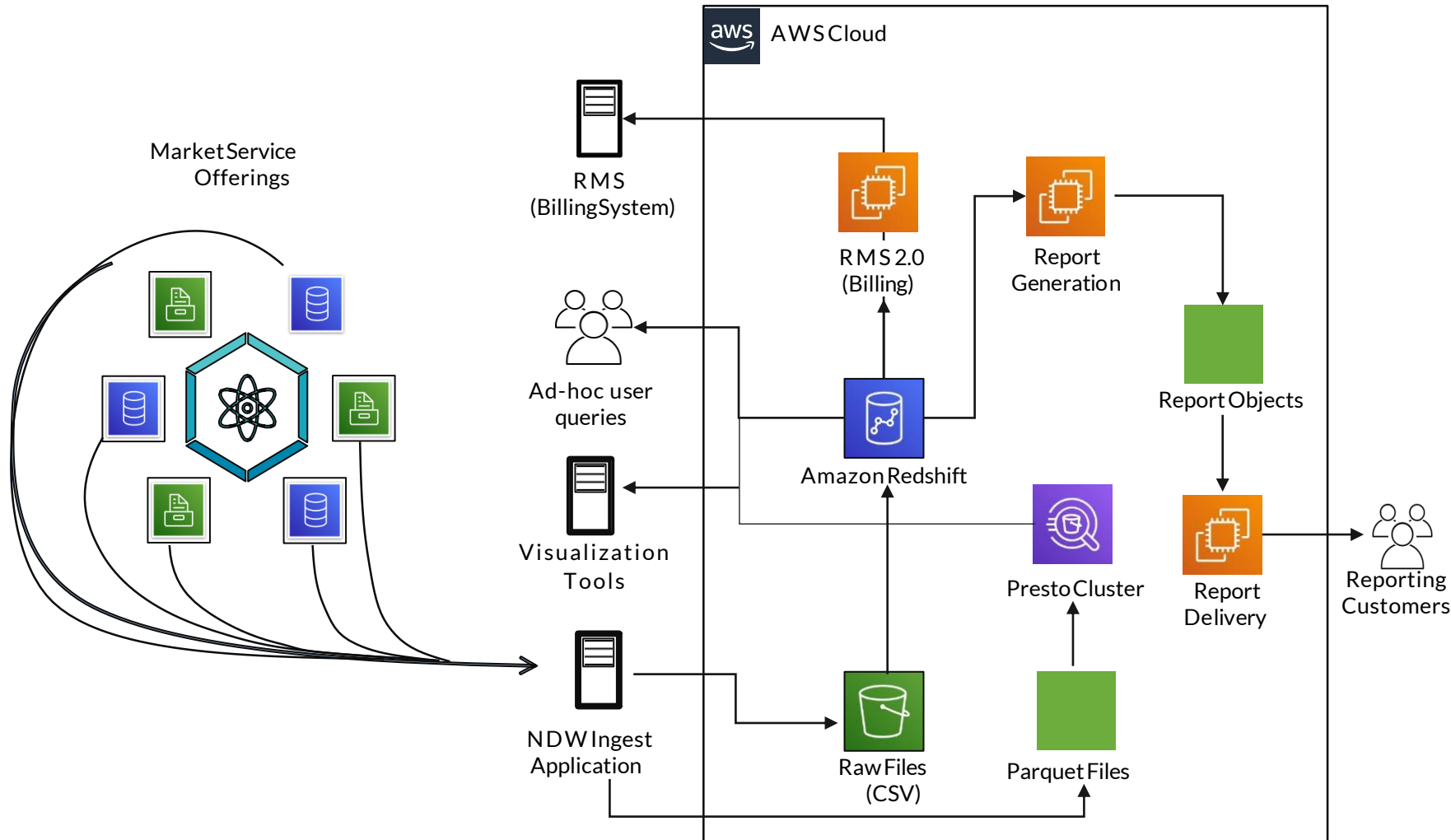
## About Nasdaq

- Lists more than 3,500 companies in 35 countries, representing more than **\$8.8tn** in total market value
- Leading index provider with **41,000+** indexes across asset classes and geographies
- Over **10,000 corporate clients in 60 countries**
- **100+ Data product offerings** supporting \$2.5+mn investment professionals and users IN 98 countries
- Nasdaq technology powers over 70 MARKETPLACES, regulators, CSDs and clearing-houses in over 50 COUNTRIES

## Technology Challenges and Key Drivers for Change (2013)

- Expensive on-premise data warehousing and billing systems (RMS), costing ~\$1.16mn annually
- Limited storage and compute capacity
- Ever increasing data from orders, trades, and quotes (~4-6 bn rows inserted per trading day)
- Difficult to manage and maintain the hardware assets
- Service-level agreements required overnight for batch processes to complete prior to the next morning

# CASE STUDY: Nasdaq



## Nasdaq Data Warehouse 1.0 - 2013

Migrated on-premise DW to Redshift

### Impact

- Costs reduced to 43% of the on-prem budget for same data set
- Handled increasing data (~14bn rows/day) at tremendous write speed (~2.76mn rows/second)
- Tuned queries running faster than the legacy on-premise data warehouse system
- Able to scale seamlessly to meet future needs



# SESSION SUMMARY

Redshift is used by 10,000+ organizations in the world for their data warehousing needs, across various industries

Redshift scales seamlessly, both vertically and horizontally to cater increase in workload

Traditional data warehouses have many limitations in term of scalability, cost, performance and variety of data they can handle

Redshift supports industry standard ANSI SQL

Redshift is ~3 times faster than other traditional data warehouses in the market

Redshift has a pay for what you use model and is comparatively cheaper than other cloud data warehouses.

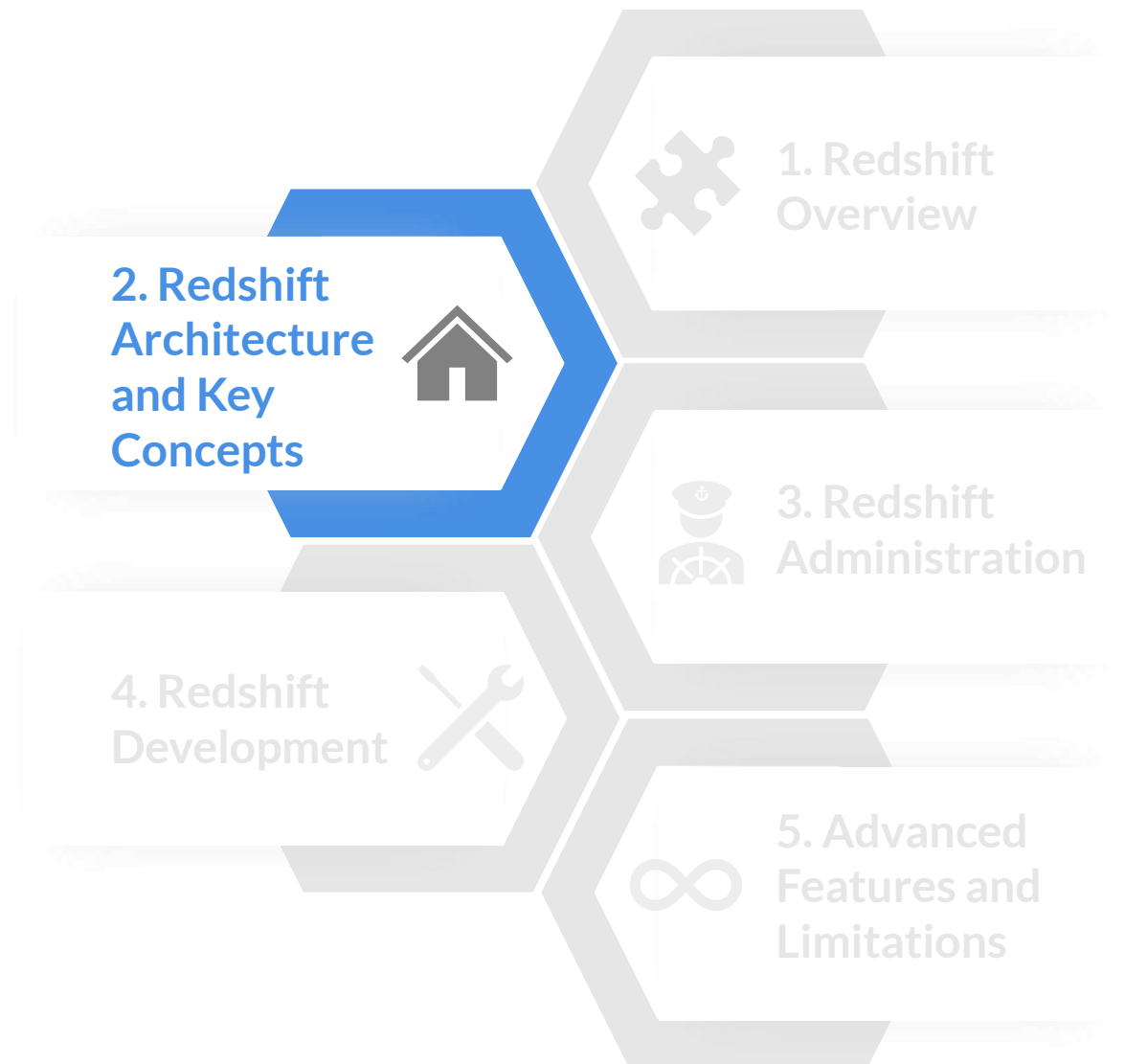
Since it is offered as PaaS, it is fully managed by Amazon.

Redshift has built-in security which makes it the most preferred DW solution

## Session 2: Redshift Architecture and Key Concepts

Upon completion of this session, you will be able to answer the following questions:

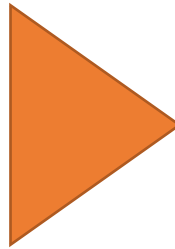
- What is Amazon Redshift?
- How does the architecture of Amazon Redshift look like?
- What is MPP?
- What are Sort Keys and Distribution Keys?
- What is Compression Encodings and why are these useful?
- What is Concurrency Scaling?



# WHAT IS AMAZON REDSHIFT?

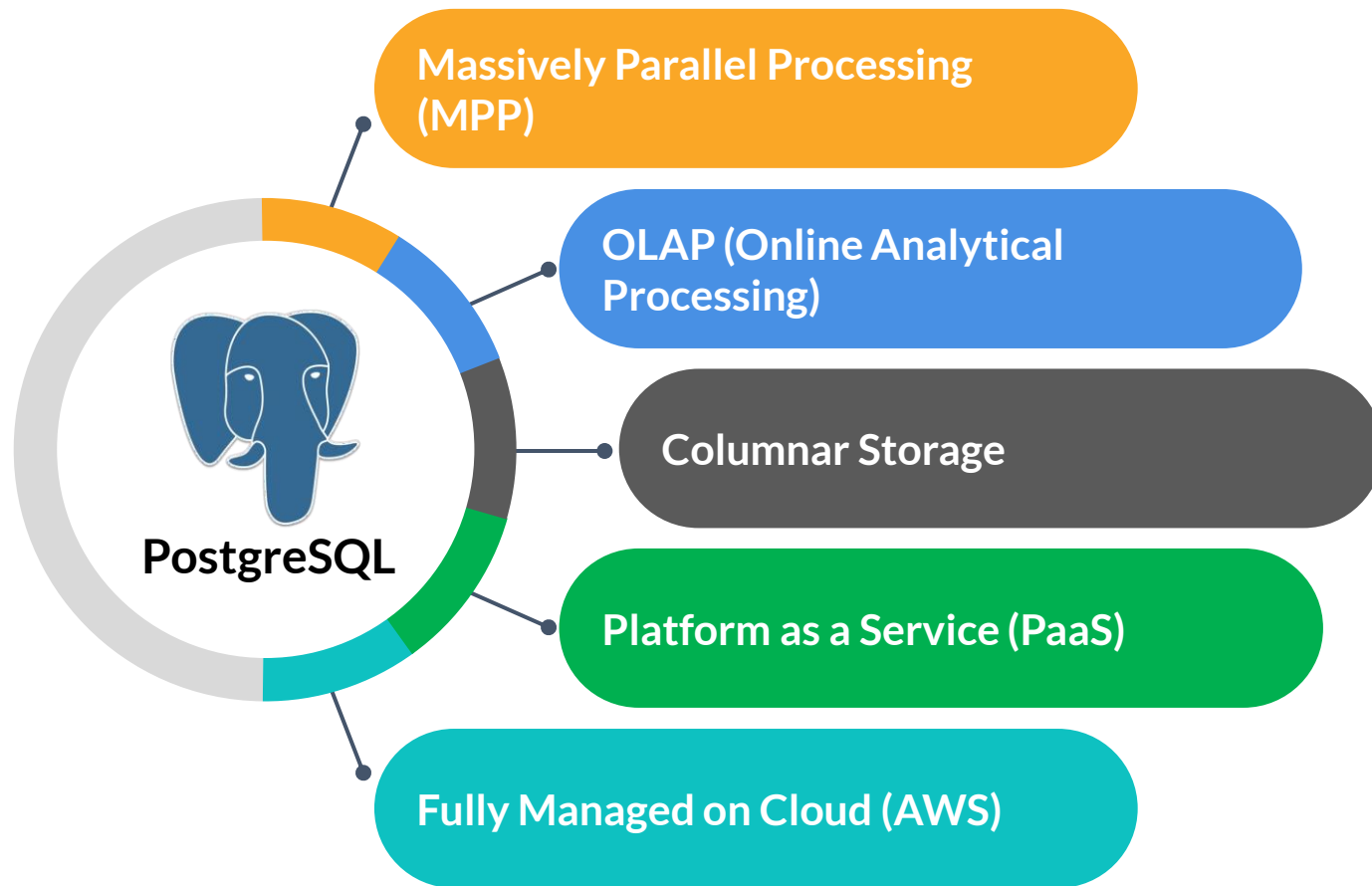


AMBASSADOR



AMBIEROD

# WHAT IS AMAZON REDSHIFT?



## AMAZON REDSHIFT



Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Amazon Redshift delivers 10 times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on a high-performance disk.

# REDSHIFT ARCHITECTURE

**Shared nothing, Massively parallel architecture**

## Leader node

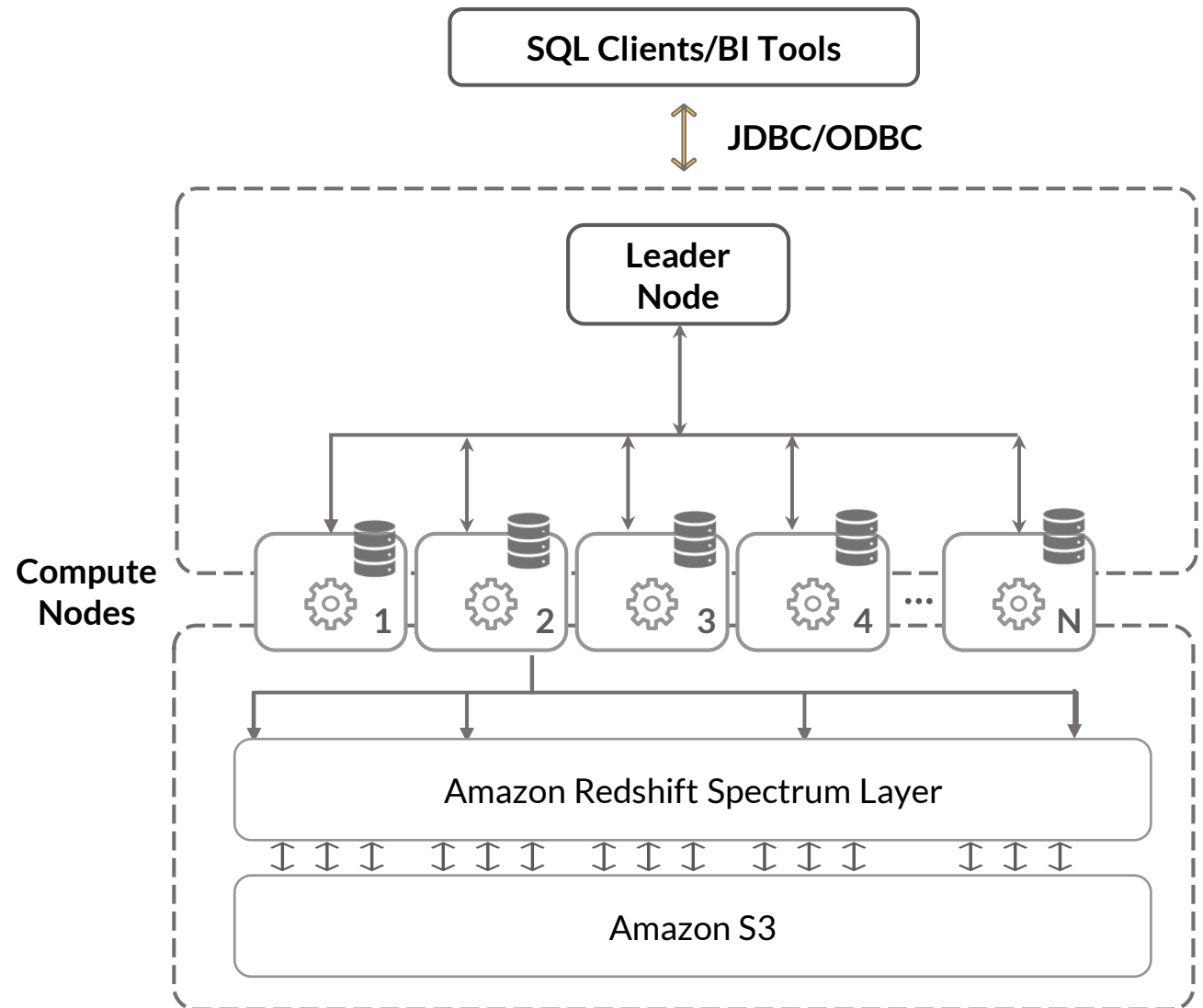
- SQL endpoint
- Stores metadata
- Coordinates parallel SQL processing

## Compute nodes

- Local, columnar storage
- Executes queries in parallel
- Load, unload, backup and restore

## Amazon Redshift Spectrum nodes

- Execute queries directly against
- Amazon Simple Storage Service (Amazon S3)



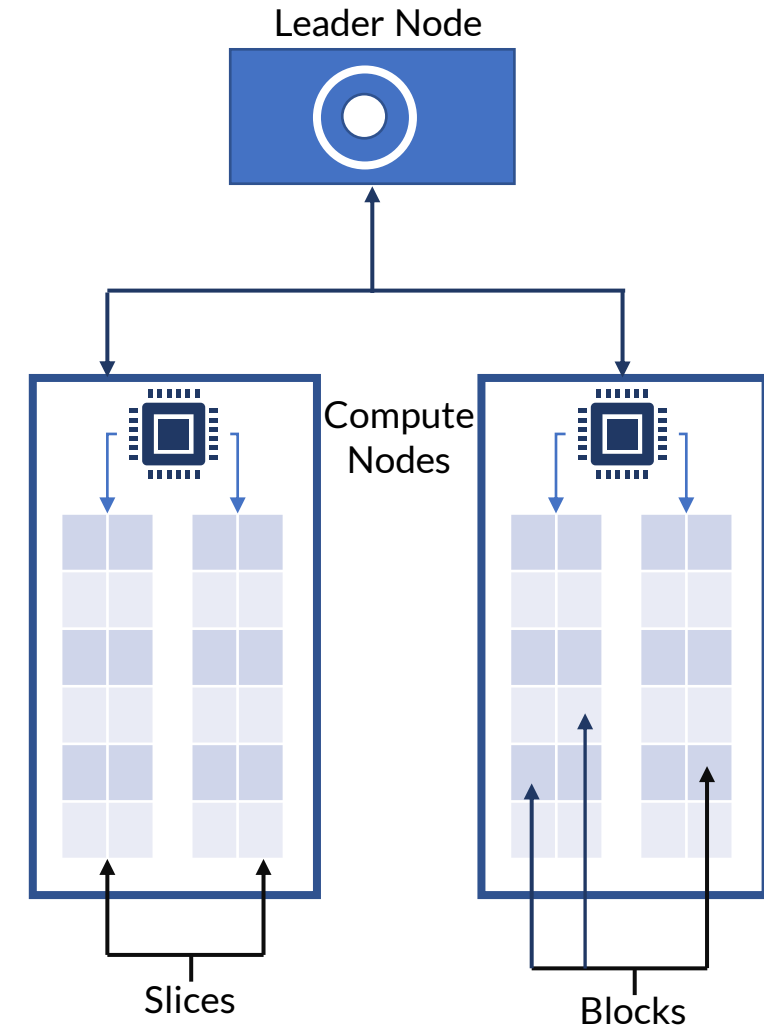
# SLICES AND BLOCKS

## Slices

- Each compute node is partitioned into 2 or 16 slices, which is determined by the node type
- A slice can be thought of like a virtual compute node (within a compute node); each slice is allocated a portion of the node's memory and disk space.
- Table rows (the actual data) are distributed to slices

## Blocks

- Slices are further divided into blocks (1 MB each)
- A full block can contain millions of values
- Each block can be encoded/compressed with one of the 13 encodings available



# COLUMNAR ARCHITECTURE

Amazon Redshift uses a columnar architecture for storing data on disk

## Goals

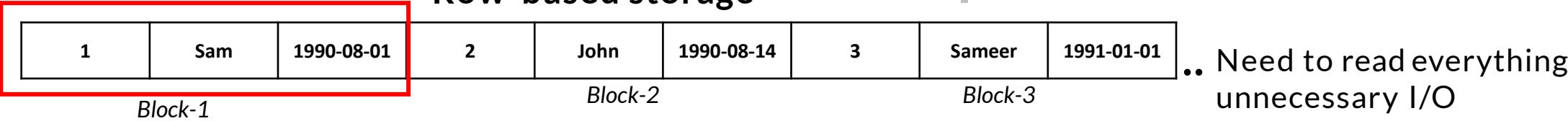
- Reduce I/O for analytics queries
- Only read the required column data

```
CREATE TABLE rs_tbl (  
  id      INT  
,name    CHAR(30)  
,dob     DATE  
);
```

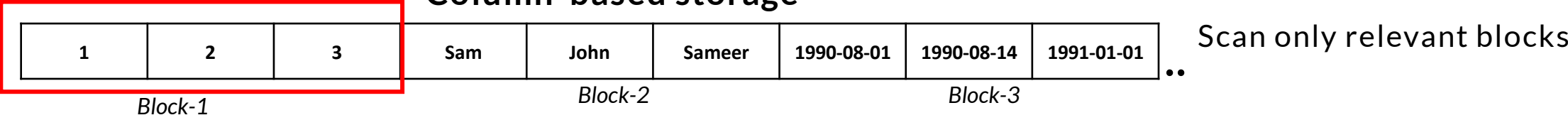
```
SELECT min(dob) FROM rs_tbl;
```

id	loc	dob
1	Sam	1990-08-01
2	John	1990-08-14
3	Sameer	1991-01-01

## Row-based storage

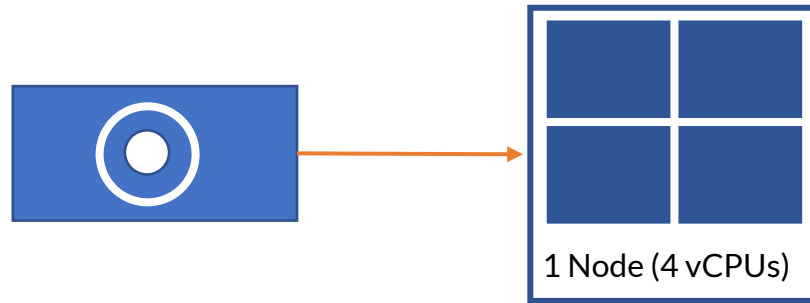


## Column-based storage

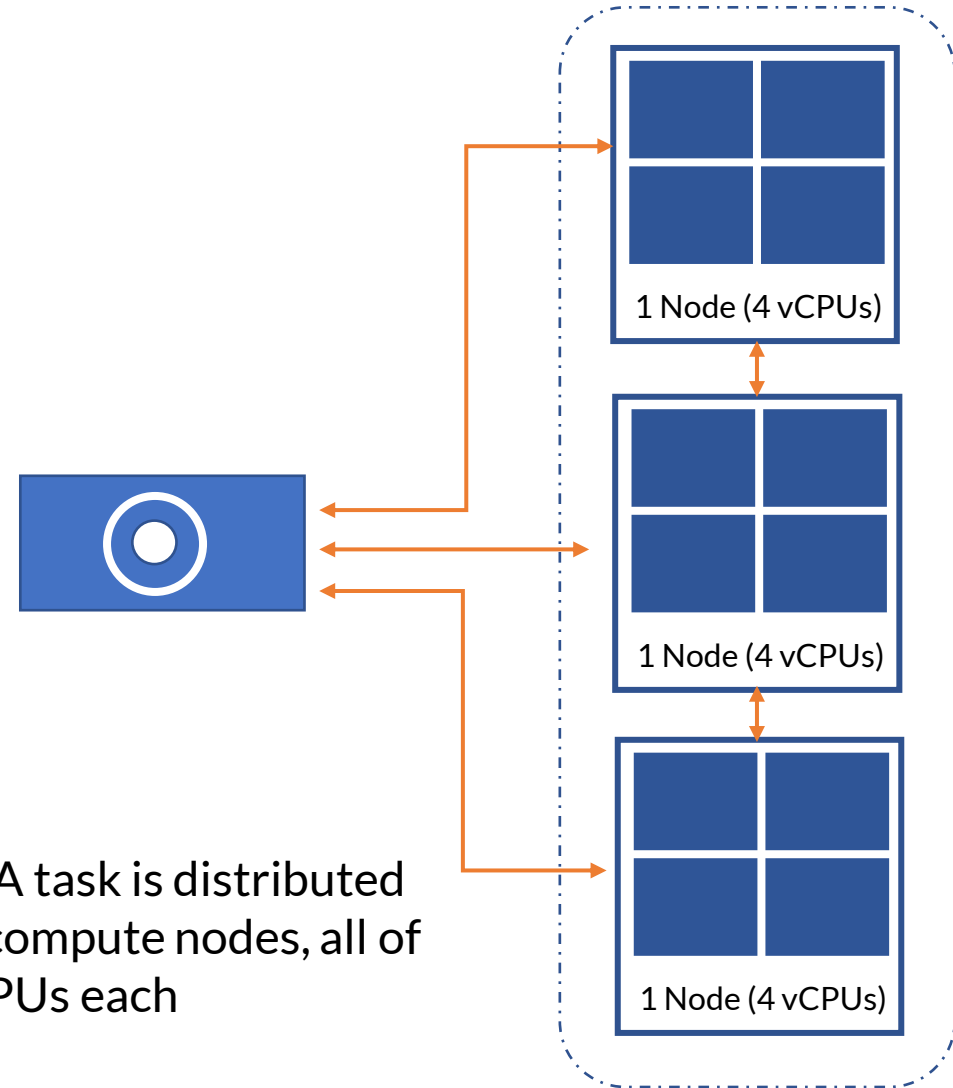


# MASSIVELY PARALLEL PROCESSING (MPP)

Single Node: A task can only be executed on one node, which has four CPUs.

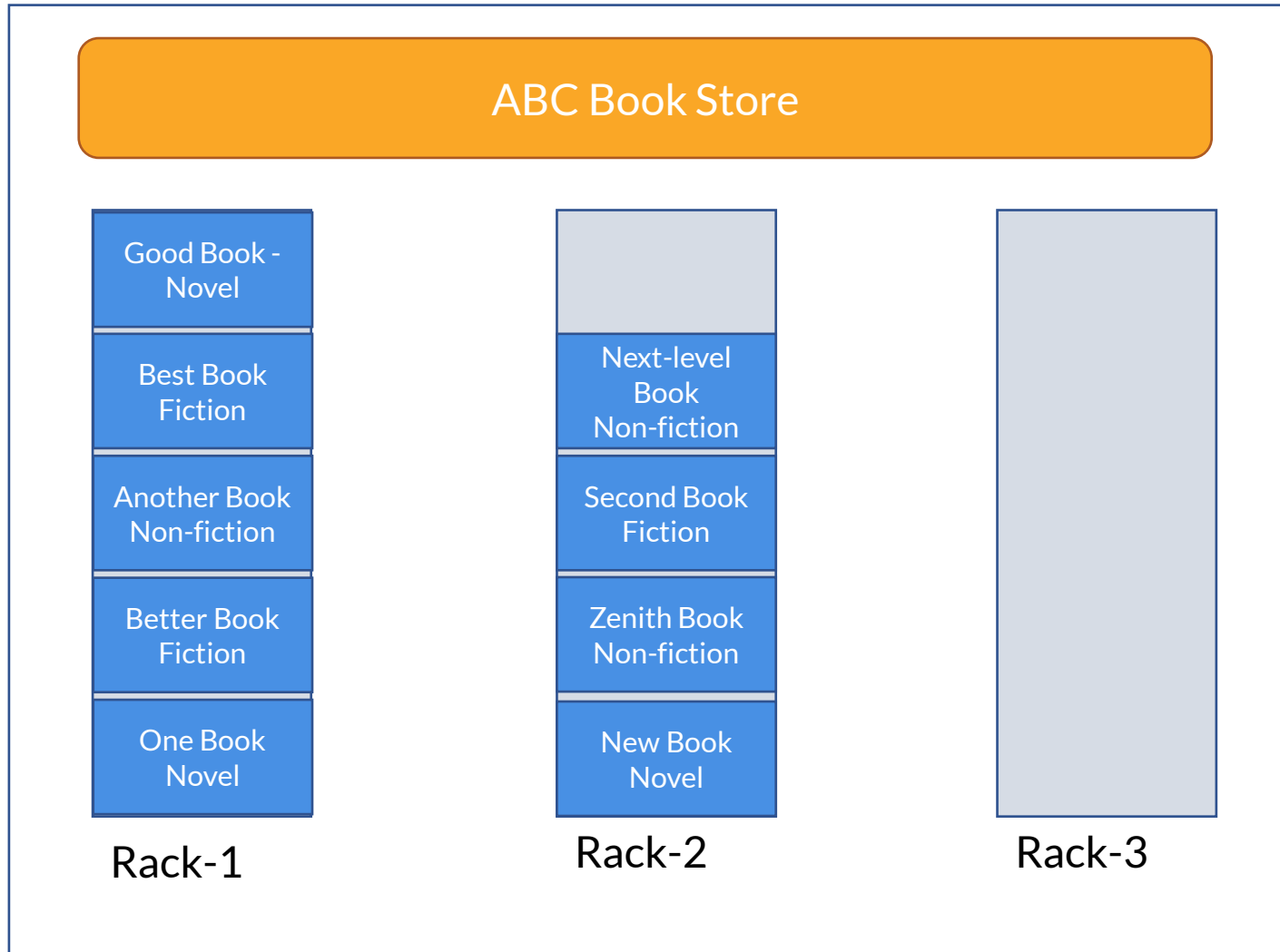


Multiple Nodes: A task is distributed among multiple compute nodes, all of which have 4 vCPUs each





# SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



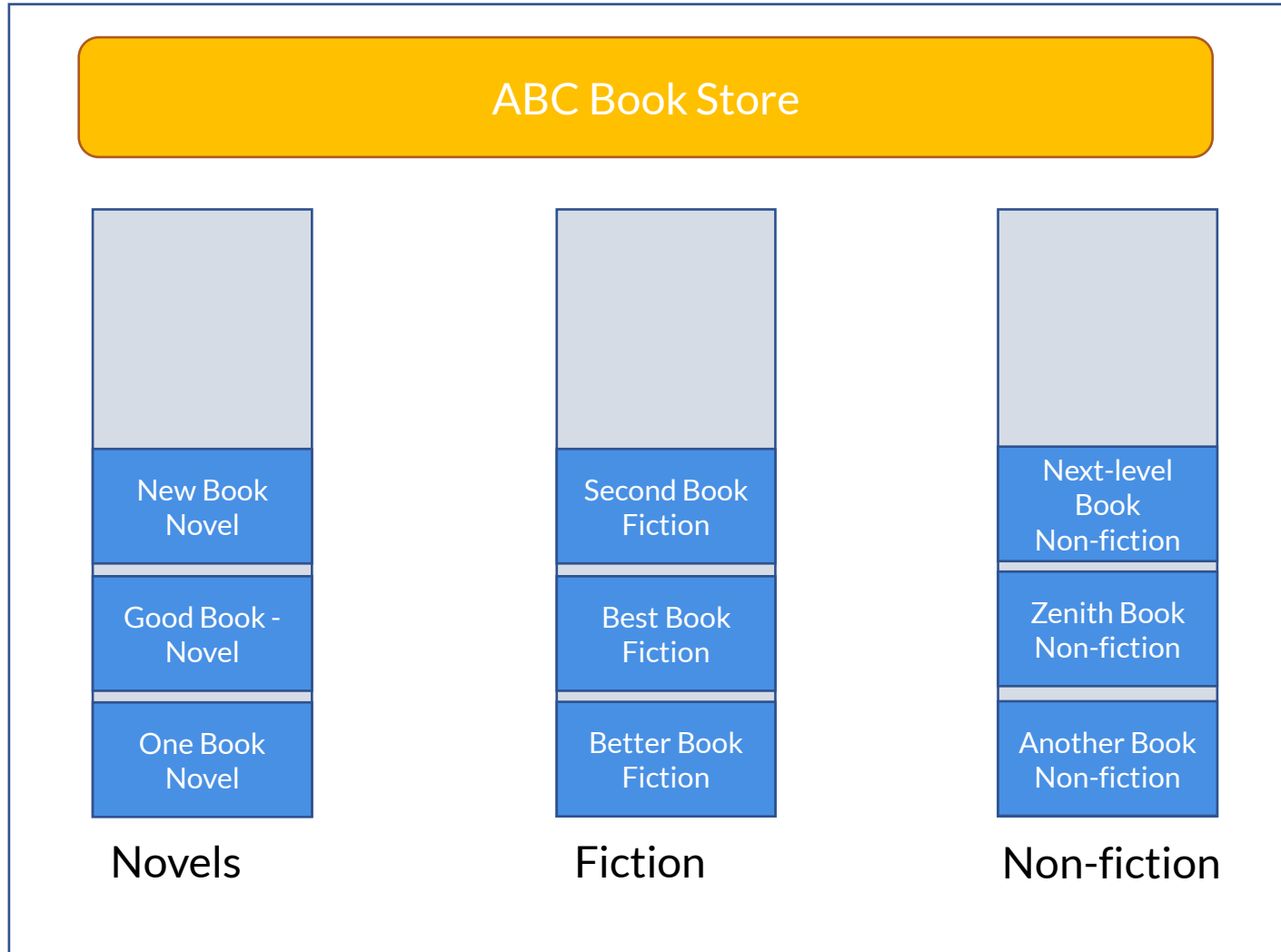
ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 1: First Book goes on rack-1, followed by second Book, and so on...Once rack-1 is full, only then they move to rack-2

Problem Statement:  
How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

**HARD**

# SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



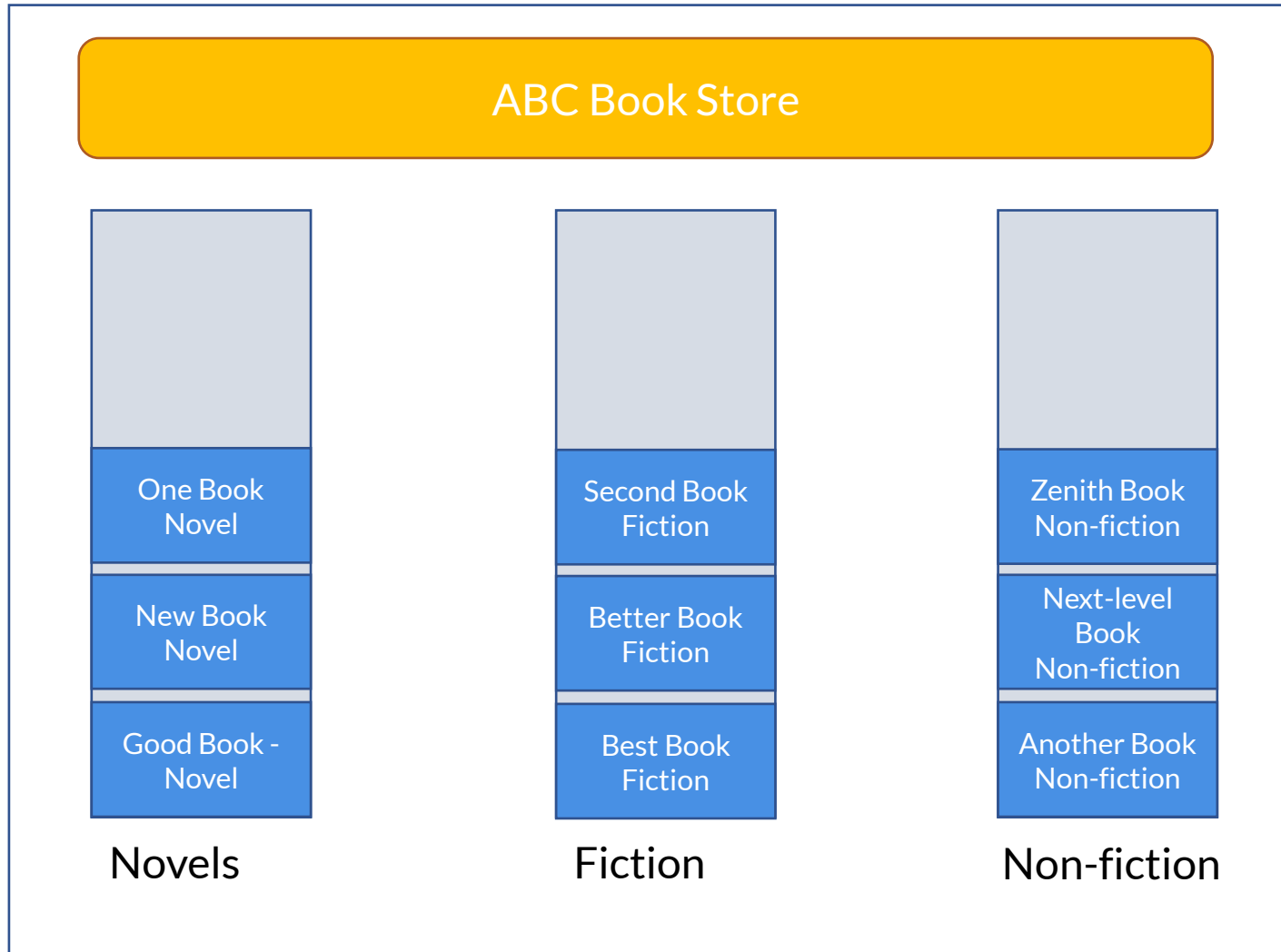
ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 2: They name the shelves according to the Book 'genres' and arrange each Book in the respective genre rack

Problem Statement:  
How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

**EASIER**

# SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 3: Along with arranging the Book in different racks based on their 'genre', They arrange the Books in alphabetical order, according to their 'Title' within the racks.

Problem Statement:

How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

**EASIEST**

# SORT KEYS AND ZONE MAPS

## Goal

- Make queries run faster by increasing the effectiveness of zone maps and reducing I/O

## Impact

- Enables range-restricted scans to prune blocks by leveraging zone maps

## Zone Maps:

- In-memory block metadata
- Track the minimum and maximum value for each block.
- Effectively prunes blocks that do not contain data for a given query

**SELECT \* FROM RS\_TABLE WHERE DATE = '09-JUNE-2020'**

### UNSORTED TABLE



MIN: 01 JUN 2020  
MAX: 20 JUN 2020



MIN: 08 JUN 2020  
MAX: 30 JUN 2020



MIN: 12 JUN 2020  
MAX: 20 JUN 2020



MIN: 02 JUN 2020  
MAX: 25 JUN 2020

### SORTED TABLE



MIN: 01 JUN 2020  
MAX: 06 JUN 2020



MIN: 07 JUN 2020  
MAX: 12 JUN 2020



MIN: 13 JUN 2020  
MAX: 18 JUN 2020



MIN: 18 JUN 2020  
MAX: 25 JUN 2020

# TYPES OF SORT KEYS

## Single Column

**SORTKEY (Date)**

Date	Region	Country
2-JUNE-2020	OCEANIA	NEW ZEALAND
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	AFRICA	ZAMBIA
3-JUNE-2020	ASIA	HONGKONG
3-JUNE-2020	EUROPE	GERMANY
3-JUNE-2020	ASIA	KOREA

- Best for queries that use the 1st column as primary filter
- Can speed up joins and group by statements

## Compound

**SORTKEY COMPOUND (Date, Region)**

Date	Region	Country
2-JUNE-2020	AFRICA	ZAMBIA
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	OCEANIA	NEW ZEALAND
3-JUNE-2020	ASIA	HONGKONG
3-JUNE-2020	ASIA	KOREA
3-JUNE-2020	EUROPE	GERMANY

- Table sorted by order as given in sort key
- Best for queries that use the 1st column as primary filter, then others

## Interleaved

**SORTKEY INTERLEAVED (Date, Region, Country)**

Date	Region	Country
2-JUNE-2020	OCEANIA	NEW ZEALAND
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	AFRICA	ZAMBIA
3-JUNE-2020	ASIA	HONGKONG
3-JUNE-2020	EUROPE	GERMANY
3-JUNE-2020	ASIA	KOREA

- Equal weight given to each column
- Best for queries that use different filter columns

# DATA DISTRIBUTION

**Distribution style** defines how the data will be stored on the compute nodes.

**KEY:** Value is hashed, same value goes to same location (slice)

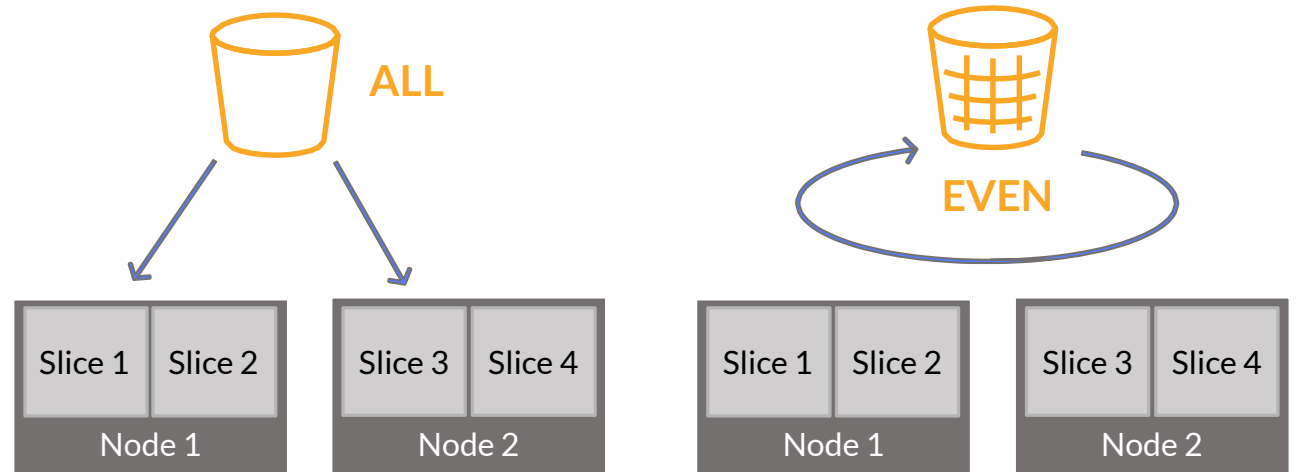
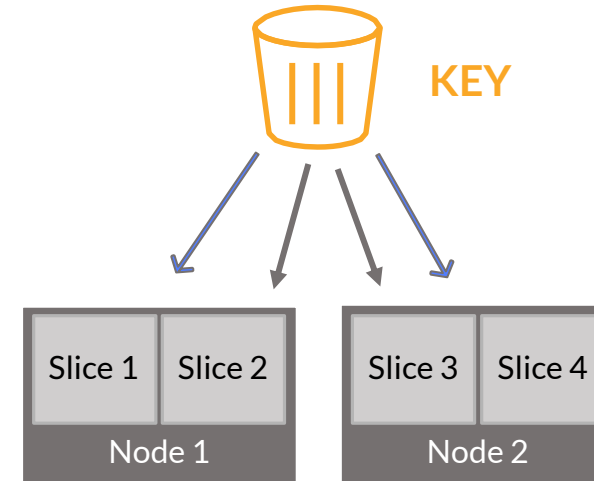
**ALL:** Full table data goes to the first slice of every node

**EVEN:** Round robin

**AUTO:** ALL, followed by EVEN

**Goals:**

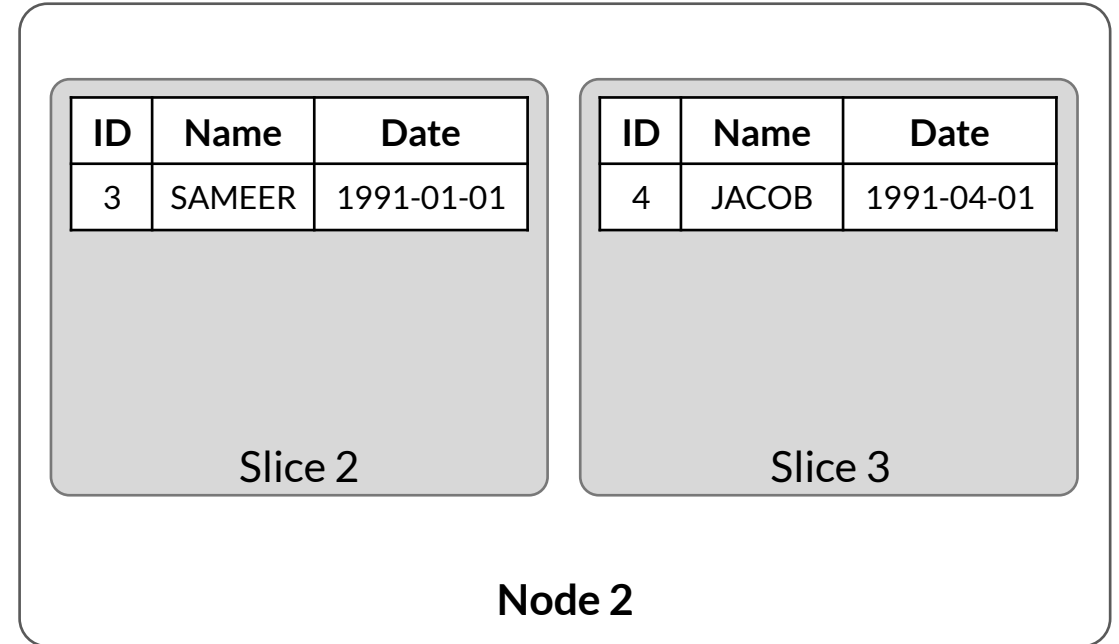
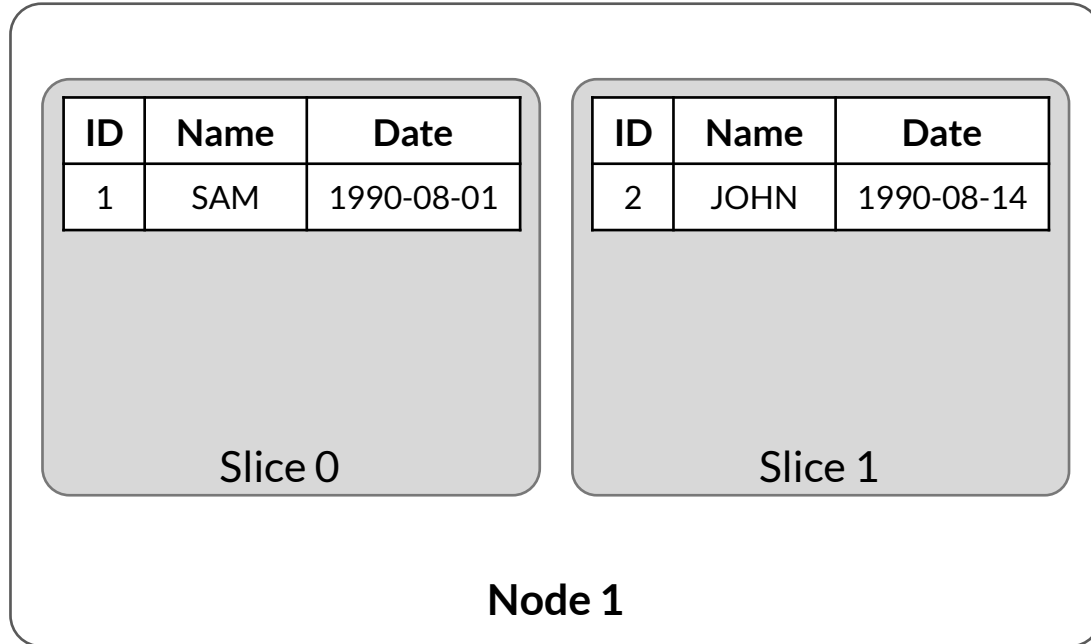
- Distribute data evenly for parallel processing
- Minimise data movement during query processing



# DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (  
    id      INT,  
    name    VARCHAR(20),  
    dob     DATE  
) DISTSTYLE EVEN;
```

```
INSERT INTO RS_DEMO VALUES  
(1, 'SAM', '1990-08-01'),  
(2, 'JOHN', '1990-08-14'),  
(3, 'SAMEER', '1991-01-01'),  
(4, 'JACOB', '1991-04-01');
```



# DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (  
    id      INT,  
    name    VARCHAR(20),  
    dob     DATE  
) DISTSTYLE KEY DISTKEY (name);
```

```
INSERT INTO RS_DEMO VALUES  
(1, 'SAM', '1990-08-01'),  
(2, 'JOHN', '1990-08-14'),  
(3, 'SAMEER', '1991-01-01'),  
(4, 'JACOB', '1991-04-01');
```

ID	Name	Date
1	SAM	1990-08-01
3	SAMEER	1991-01-01

Slice 0

ID	Name	Date
2	JOHN	1990-08-14
4	JACOB	1991-04-01

Slice 1

Node 1



Slice 2



Slice 3

Node 2



# DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (  
    id      INT,  
    name    VARCHAR(20),  
    dob     DATE  
) DISTSTYLE ALL;
```

```
INSERT INTO RS_DEMO VALUES  
(1, 'SAM', '1990-08-01'),  
(2, 'JOHN', '1990-08-14'),  
(3, 'SAMEER', '1991-01-01'),  
(4, 'JACOB', '1991-04-01');
```

ID	Name	Date
1	SAM	1990-08-01
2	JOHN	1990-08-14
3	SAMEER	1991-01-01
4	JACOB	1991-04-01

Slice 0



Slice 1

Node 1

ID	Name	Date
1	SAM	1990-08-01
2	JOHN	1990-08-14
3	SAMEER	1991-01-01
4	JACOB	1991-04-01

Slice 2



Slice 3

Node 2

# COMPRESSION ENCODINGS

## Goals

- Allow more data to be stored within an Amazon Redshift cluster
- Improve query performance by decreasing I/O
- 13 different compression encodings available

## Impact

- Allows storage of 2–4 times more data within the cluster

By default, **COPY** automatically analyses and compresses data on first load into an empty table

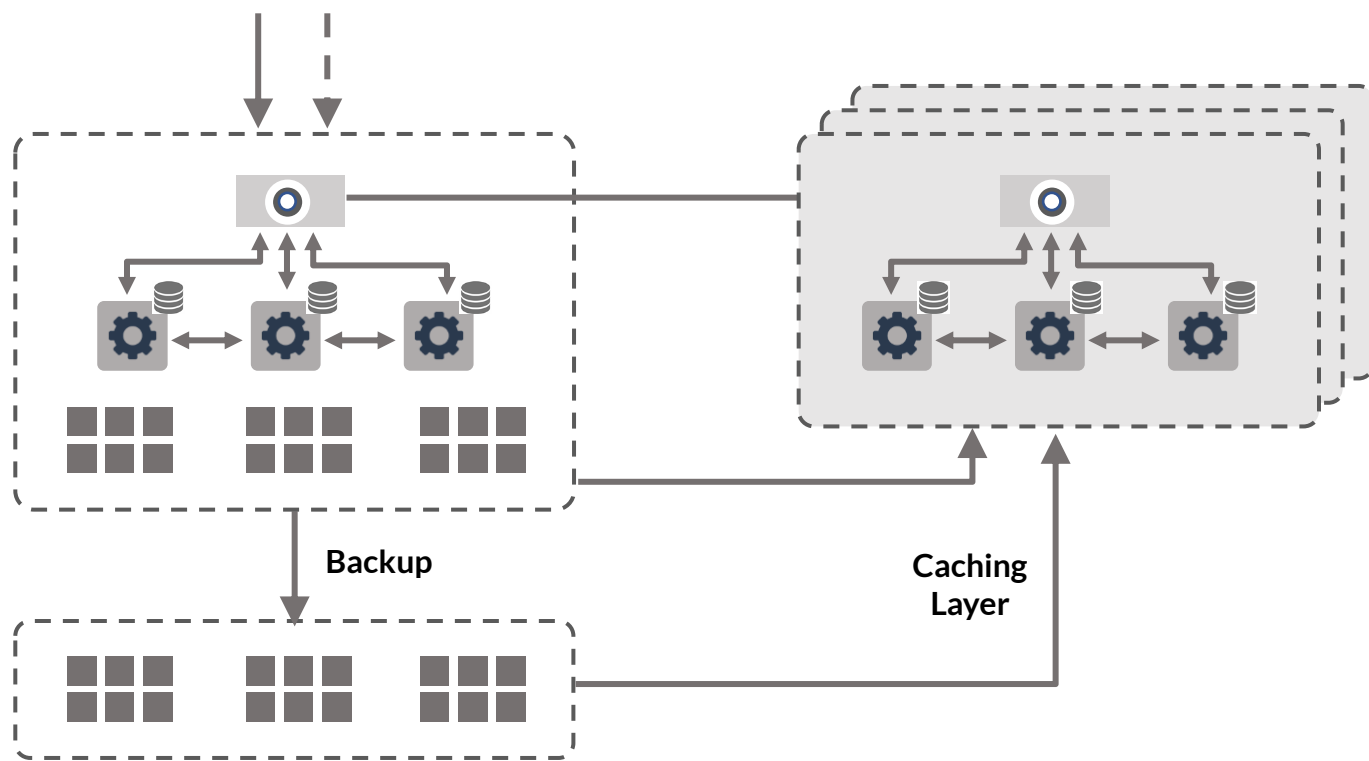
**ANALYZE COMPRESSION** is a built-in command that finds the optimal compression for each column in an existing table

**AZ64** is a new, Amazon proprietary compression encoding algorithm, which offers better storage savings and high query performance compared to other Amazon Redshift encodings

	AZ64 storage savings	AZ64 performance speed ups
RAW	60–70% less storage	25–30% faster
LZO	35% less storage	40% faster
ZSTD	Comparable footprint	70% faster

# CONCURRENCY SCALING

Amazon Redshift automatically adds compute capacity within seconds to serve unprecedented read queries. Thus supporting virtually unlimited number of connections to the cluster.



For every 24 hours that your main cluster is in use, you accrue a one-hour credit for Concurrency Scaling. This means that Concurrency Scaling is free for >97% of customers.

How it works:

- 1 All queries go to the leader node
- 2 When queries begin queuing, Amazon Redshift creates a snapshot of the cluster
- 3 Concurrency scaling cluster is provisioned with the help of snapshot
- 4 More capacity is added to the concurrency scaling cluster as needed

# SESSION SUMMARY

Redshift is based on the Massively Parallel Processing architecture

In Redshift, data is stored in columnar format, which offers superior performance for analytical queries

Sort keys determines the order in which data on slices will be sorted

Distribution style for a table defines how and where the data will be physically stored on the slices.

Redshift does not have the concept of Indexes like in traditional databases, instead it has zone maps

Individual columns in Redshift tables can be compressed using Compression Encodings. There are 13 different encodings available

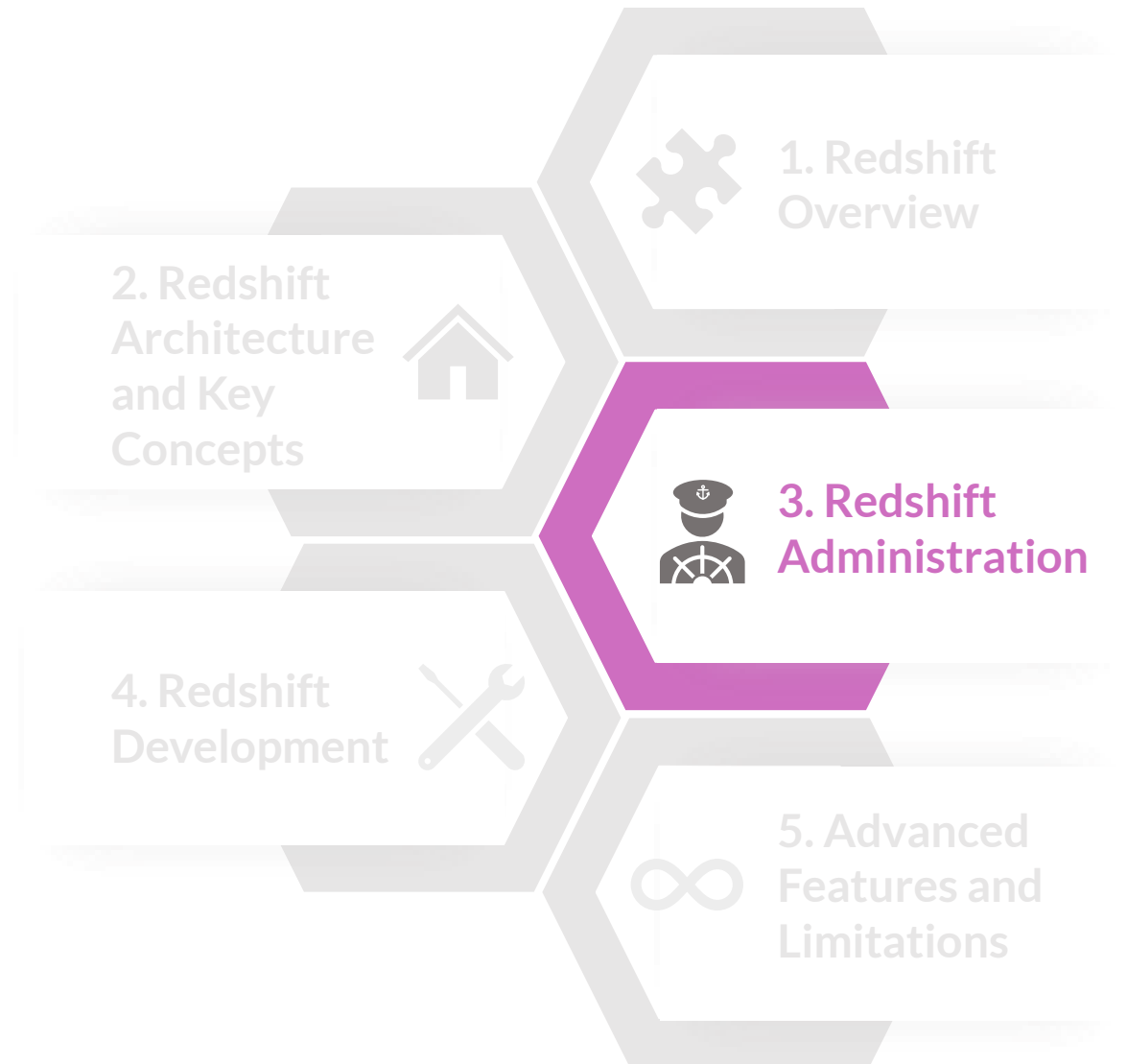
Redshift does not enforce constraints such as unique key, primary key and referential integrity, but the optimizer uses it to build the query plan

Since there is some overhead (sorting and distribution) when writing rows into table, Redshift is not suitable for OLTP workloads

## Session 3: Redshift Administration

Upon completion of this session, you will learn:

1. How to create a new Redshift cluster?
2. What are the different node types available for Redshift?
3. What is Workload Management (WLM)?
4. How to resize a Redshift cluster?
5. How to take backup of and restore a Redshift cluster?
6. What security measures are in place for Amazon Redshift?



# CREATING A REDSHIFT CLUSTER

Demo

This screenshot shows the AWS Management Console for the Northern Virginia region. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The left sidebar contains navigation icons for Dashboard, Clusters, Queries, Editor, Config, Marketplace, Advisor, Alarms, and Events. The main content area displays the 'Northern Virginia' overview with several metrics:

- OVERVIEW:** 1 Cluster, 2 Total nodes, 2 On-demand nodes.
- RESERVED NODES:** 0 Reserved nodes, 0 Reserved nodes used, 0 Reserved nodes available.
- SNAPSHOTS:** 2 Snapshots, 2 Automated, 0 Manual.

Below these metrics is a 'Cluster overview' section with a status bar showing 0 Available, 0 Modifying, 1 Paused, 0 Unavailable, and 0 In Maintenance. A graph at the bottom shows the number of queries over time. On the right, there is an 'Alarms' section with a 'View CloudWatch dashboard' link and a status of 0/0 In alarm.

This screenshot shows the 'Amazon Redshift > Clusters' page in the AWS Management Console. The top navigation bar and left sidebar are consistent with the previous screenshot. The main content area displays a list of clusters:

Cluster	Status	Storage capacity us...	CPU utilization	Snaph...	Notificati...	Tags
redshift-cluster-1	Paused	dc2.large   2 nodes   320 GB		2 snapshots		

At the top right of the cluster list, there is a 'Create cluster' button. Below the list, there is a 'Query cluster' button and an 'Actions' dropdown menu.

This screenshot shows the 'Create cluster' wizard in the AWS Management Console. The top navigation bar and left sidebar are consistent with the previous screenshots. The main content area displays the 'Create cluster' page with the following sections:

- Cluster identifier:** A text box containing 'redshift-cluster-2'. Below it, a note states: 'The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).'
- Node type:** A section titled 'Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.' It contains two main categories: 'RA3' (High performance with scalable managed storage) and 'DC2' (High performance with fixed local SSD storage).

The 'RA3' section is highlighted with a 'Recommended' badge and contains two options:

- ra3.4xlarge:** Managed storage: up to 64 TB/node. Price: \$3.26/node/hour, \$0.024/GB/month.
- ra3.16xlarge:** Managed storage: up to 64 TB/node. Price: \$13.04/node/hour, \$0.024/GB/month.

The 'DC2' section contains two options:

- dc2.large:** Storage: 160 GB/node. Price: \$0.25/node/hour.
- dc2.8xlarge:** Storage: 2.6 TB/node. Price: \$4.80/node/hour.

At the bottom, there is a 'Storage' section with a dropdown menu set to 'SSD' and a selection of 'dc2.large'.

# REDSHIFT NODE TYPES

## Dense storage— DS2

- For large data warehouses with HDD (Magnetic) disks
- For ds2.xlarge \$0.85 per Hour
- For ds2.8xlarge \$6.80 per Hour

## Dense compute— DC2

- For compute-intensive data warehouses with Solid-state disks
- For dc2.large \$0.25 per Hour
- For dc2.8xlarge \$4.80 per Hour

## Amazon Redshift analytics—RA3

- Amazon Redshift Managed Storage (RMS)—Solid-state disks + Amazon S3
- For ra3.4xlarge \$3.26 per Hour
- For ra3.16xlarge \$13.04 per Hour

Instance type	Disk type	Size	Memory	CPUs	Slices
RA3 4xlarge (new)	RMS	Scales to 16 TB	96 GB	12	4
RA3 16xlarge (new)	RMS	Scales to 64 TB	384 GB	48	16
DC2 large	SSD	160 GB	16 GB	2	2
DC2 8xlarge	SSD	2.56 TB	244 GB	32	16
DS2 xlarge	Magnetic	2 TB	32 GB	4	2
DS2 8xlarge	Magnetic	16 TB	244 GB	36	16

# REDSHIFT MAINTENANCE



Amazon Redshift turbo charges query performance with machine learning-based automatic optimisations

## VACUUM

- VACUUM removes rows that are marked as deleted and globally sort tables
- For the majority of workload, AUTO VACUUM DELETE will reclaim space and AUTO TABLE SORT will sort the needed portions of the table
- In cases where you know your workload, VACUUM can be run manually

## ANALYZE

- The ANALYZE process collects table statistics for optimal query planning
- In the vast majority of cases, AUTO ANALYZE automatically handles statistics gathering



Automatic  
Analyze



Automatic Table  
Distribution Style



Automatic  
Vacuum



Automatic  
Table Sort



Sort Key  
Advisors



# WORKLOAD MANAGEMENT

WLM allows for the separation of different query workloads

## Goals

- Prioritize important queries
- Throttle/abort less important queries
- Control concurrent number of executing queries
- Divide cluster memory
- Set query timeouts to abort long-running queries

## Queues:

- Assign a percentage of cluster memory
- SQL queries execute in queue based on
  - User group: which groups the user belongs to
  - Query group session-level variable

## Query Slot:

- Division of memory within a WLM queue, correlated with the number of simultaneous running queries
- WLM\_QUERY\_SLOT\_COUNT is a session-level variable
- Useful to increase for memory-intensive operations (e.g. large COPY, VACUUM, large INSERT INTO SELECT)

Scenario 1: Single Queue for all workloads






Scenario 2: Separate queues for separate workloads



# WORKLOAD MANAGEMENT

Demo

 Services ▾ Resource Groups ▾ 

 ta15@upgrad.com @ 0679-24... ▾ N. Virginia ▾ Support ▾

 DASHBOARD CLUSTERS QUERIES EDITOR CONFIG MARKETPLACE ADVISOR ALARMS EVENTS

Amazon Redshift &gt; Configurations &gt; Workload management &gt; Modify workload queues

Modify workload queues: redshift-param-group-01 Add queue☒ Enable **Short query acceleration** for queries whose maximum runtime is dynamic ▾ . [Learn more](#) 

### Queue 1

Memory (%)	Concurrency on main	Concurrency scaling mode	Query priority
Auto	Auto	<span>off ▾</span>	<span>High ▾</span>

#### User groups

☒ Matching wildcards

Delete

[Add user group](#)

#### Query groups

☒ Matching wildcards

Delete

[Add query group](#)

▼ Query monitoring rules (0)

[Add rule from template](#) | [Add custom rule](#)

No rules have been defined.

Default queue

# REDSHIFT SCALING – ELASTIC RESIZE

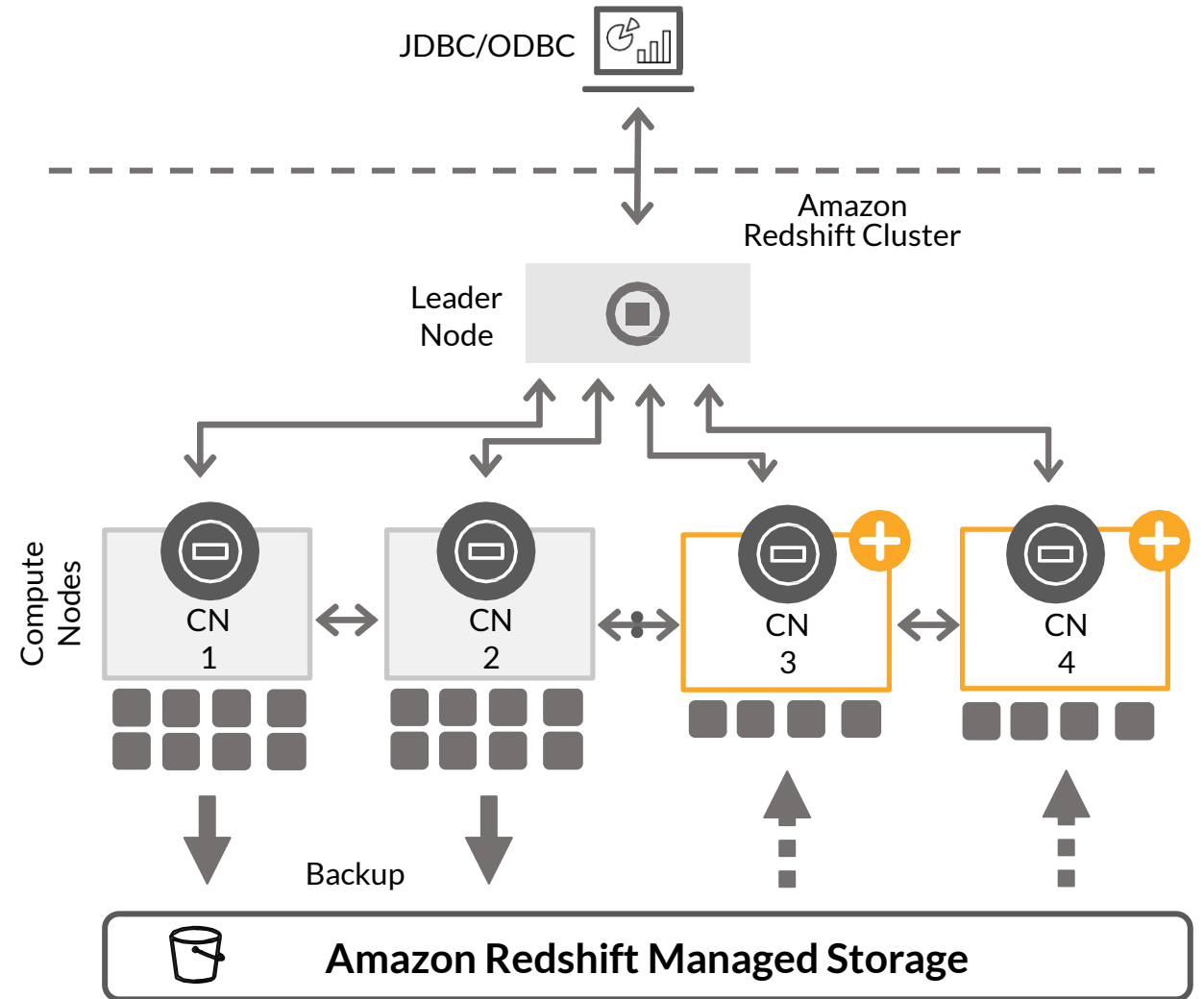
## Elastic Resize

Nodes are added to/removed from the existing cluster (within minutes)

1. As soon as the resize is requested, a backup/snapshot is taken.
2. New nodes are added and made available to the cluster immediately.
3. Data is redistributed to the node slices in the background.
4. Queries are temporarily paused and connections held open, if possible.

## Current Limitations

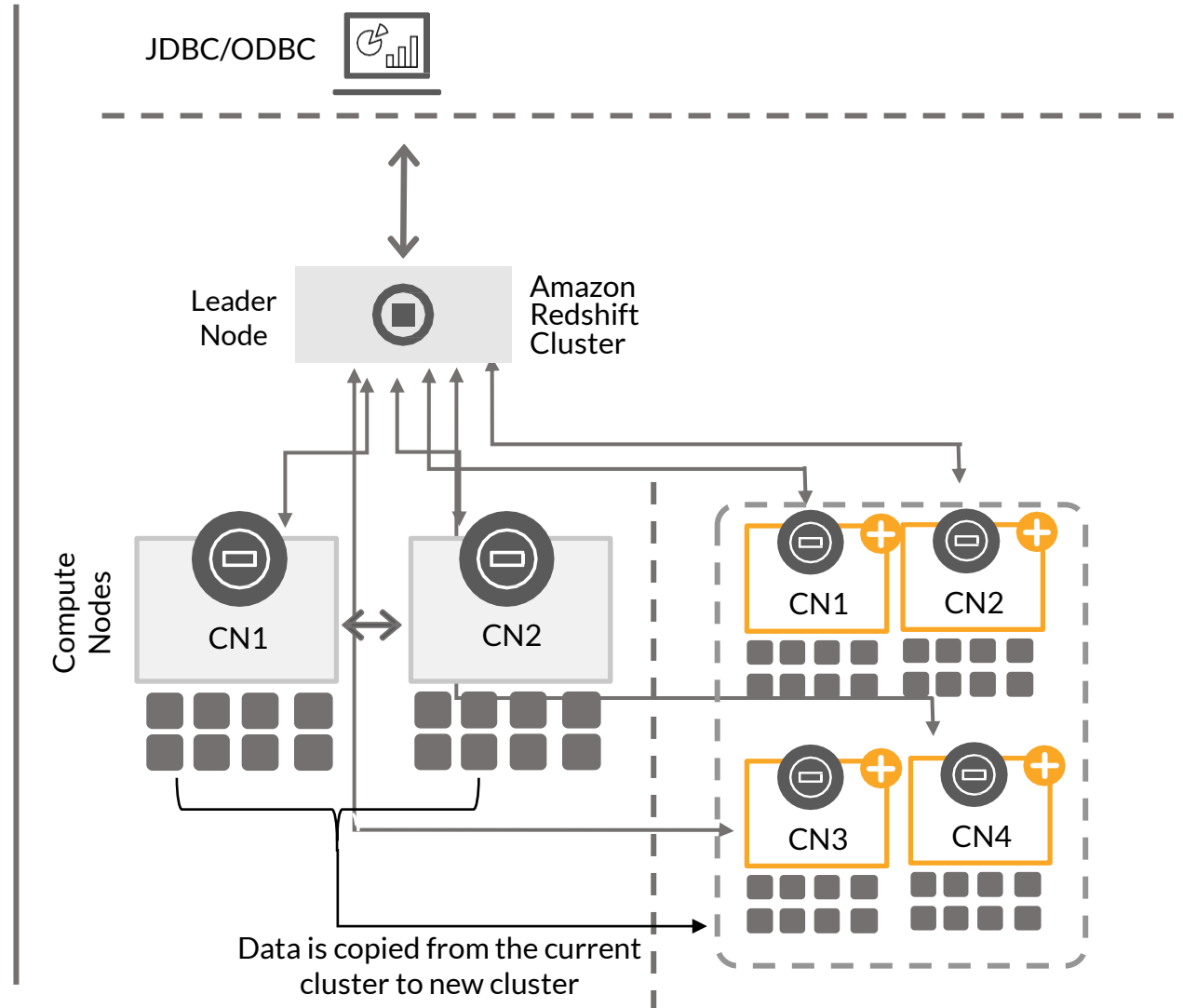
- For dc2.large or ds2.xlarge node types, you can either double or reduce **to** half the number of nodes of the original cluster, e.g., 4 to 2 or 8
- For dc2.8xlarge, ds2.8xlarge, ra3.4xlarge, or ra3.16xlarge node types, you can either reduce **till** half or double the current number of nodes, e.g., 4 to 2,3,5,6,7,8.



# REDSHIFT SCALING – CLASSIC RESIZE

## Classic Resize

1. As soon as the resize is requested, a new cluster is created in the backend
2. The original cluster goes into read-only mode, and data is copied from the original cluster to the new cluster
3. As soon as the data copy is completed, the leader node points to the new compute nodes
4. This may take a very long time, depending upon the size of data and number of nodes in the cluster



# REDSHIFT BACKUP AND RESTORE

Demo

Amazon Redshift > Clusters > Snapshots

Total snapshots: 2, Automated snapshots: 2, Manual snapshots: -, Total billable storage: 52 MB

Snapshots(0/2)

Created	Snapshot	Status	Total size	Type	Time until deletion	Cluster	Tags
4 days ago Jun 16, 2020, 4:42 PM	rs:redshift-cluster-1-2020-06-16-11-12-55	Available	52 MB	Automated	Jun 17, 2020, 4:42 PM	redshift-cluster-1 dc2.large   2 nodes	
4 days ago Jun 16, 2020, 2:06 PM	rs:redshift-cluster-1-2020-06-16-08-36-41	Available	52 MB	Automated	Jun 17, 2020, 2:06 PM	redshift-cluster-1 dc2.large   2 nodes	

Amazon Redshift > Clusters > Snapshots > Restore from snapshot

## Restore snapshot rs:redshift-cluster-1-2020-06-16-11-12-55

### Cluster configuration

**Cluster identifier**  
This is the unique key that identifies a cluster.  
redshift-cluster-1

The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

**Node type**  
Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.

#### RA3

High performance with scalable managed storage

☐ ra3.4xlarge \$3.26/node/hour  
Managed storage: \$0.024/GB/month  
up to 64 TB/node

ra3.4xlarge  
12 vCPU (gen 3)

#### DC2

High performance with fixed local SSD storage

☒ dc2.large \$0.25/node/hour  
Storage: 160 GB/node

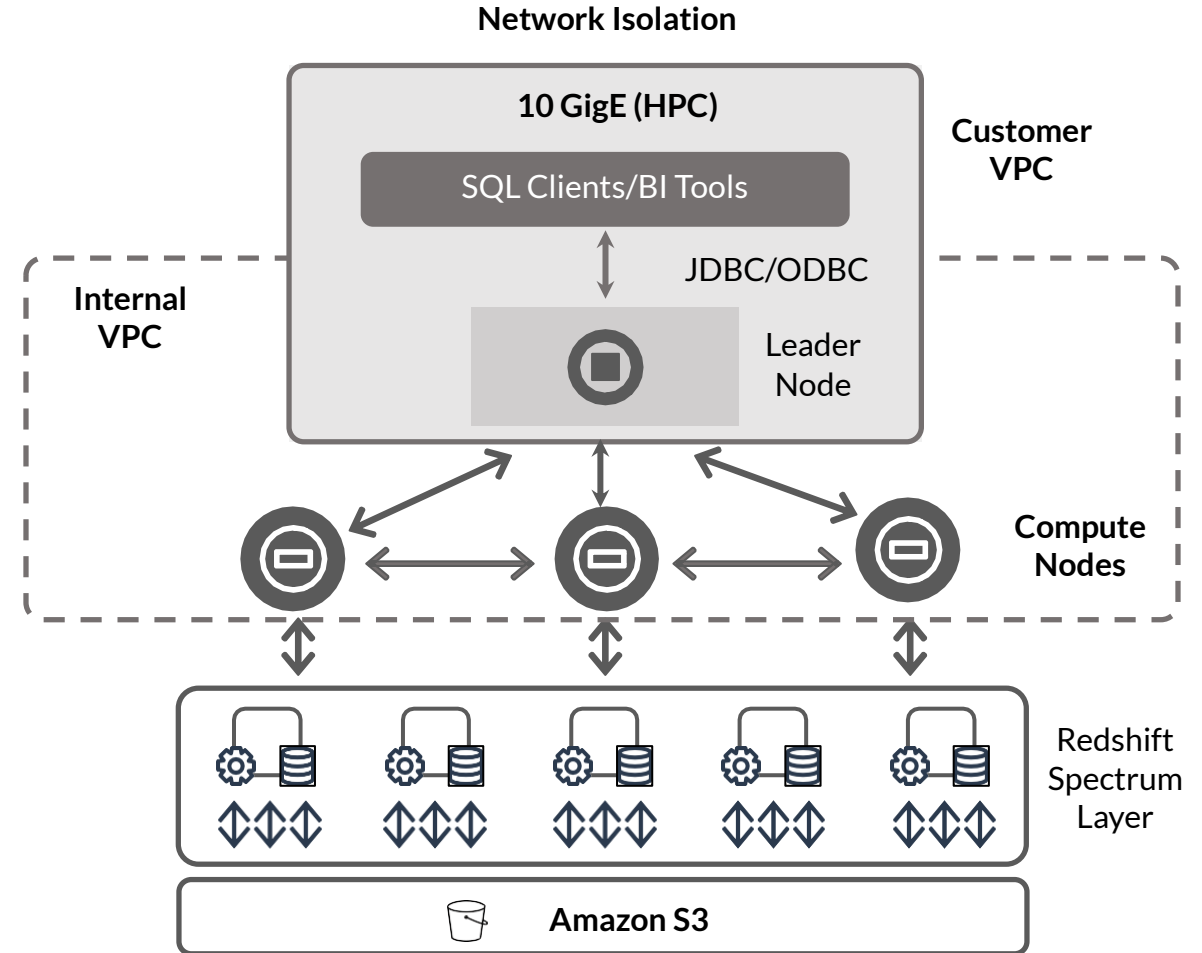
☐ dc2.8xlarge \$4.80/node/hour  
Storage: 2.6 TB/node

dc2.large  
2 vCPU (gen 2)

# FAULT TOLERANCE AND SECURITY

- SSL to secure data in transit
- Encryption to secure data at rest
  - AES-256; hardware-accelerated
  - All data blocks encrypted
- Audit logging at AWS CloudTrail integration
- Amazon VPC support
- SOC 1/2/3, PCI-DSS level 1, FedRAMP, etc.

## Select Compliance Certifications\*



# SESSION SUMMARY

You can provision Redshift clusters within minutes

The new RA3 nodes decouple storage from compute

A single node Redshift cluster has one node, which acts as Master and Compute Node

A 2-node Redshift cluster has 2 compute nodes (chargeable) and 1 master node which is offered free of cost

Redshift scales up and down within minutes through Elastic Resize

You can pause the Redshift cluster when it is not in use and resume when needed, thus saving costs

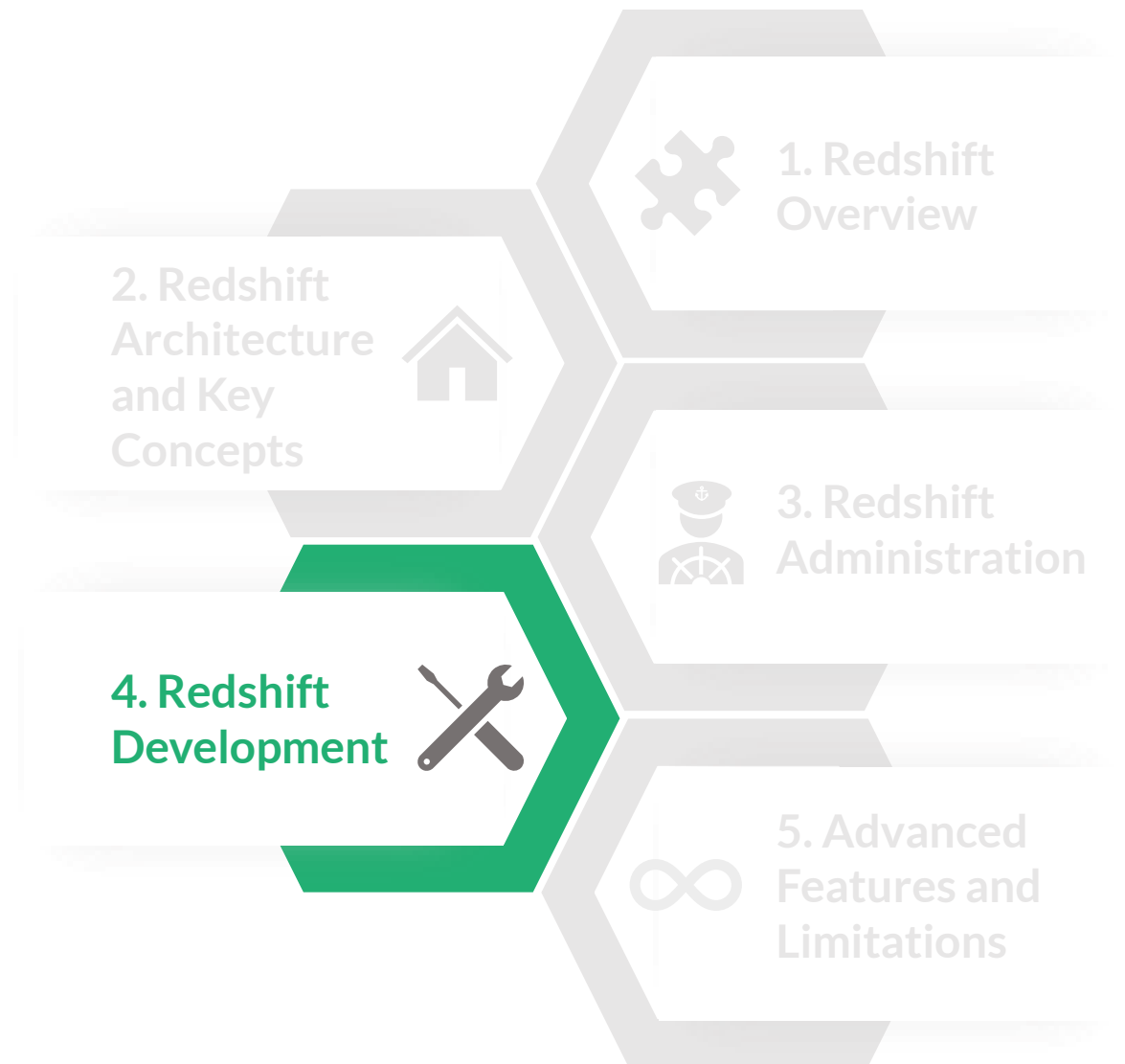
You can define different queues for different workloads through WLM

Redshift has automated snapshots (backups) which can also be scheduled.

# Session 4: Redshift Development

Upon completion of this session, you will learn:

1. How to connect and query a Redshift cluster?
2. How to create objects (database, schema, table) in Redshift?
3. What best practices we need to follow while designing a table?
4. How to load data into Redshift?
5. How to analyze data within Redshift?
6. What are stored procedures and UDFs?
7. Basics of how to tune Redshift queries?





# CONNECT TO REDSHIFT CLUSTER

Demo

The screenshot displays the AWS Redshift console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'ta15@upgrad.com' in 'N. Virginia'. The left sidebar contains navigation icons for Dashboard, Clusters, Queries, Editor (highlighted), Config, Marketplace, Advisor, Alarms, and Events. The main content area is divided into two panes. The left pane, titled 'Data objects', shows a 'Select schema' dropdown set to 'information\_schema' and a list of tables and views including 'applicable\_roles', 'check\_constraints', 'column\_domain\_usage', 'column\_privileges', 'column\_udt\_usage', 'columns', 'constraint\_column\_usage', 'constraint\_table\_usage', 'data\_type\_privileges', and 'domain\_constraints'. The right pane, titled 'Query editor', shows a connection to 'redshift-cluster-1' with a 'dev' database and 'awsuser' role. It contains a single query: 'select \* from stv\_inflight;'. Below the query editor are 'Run', 'Save', and 'Clear' buttons, along with a 'Send feedback' link. At the bottom, there are tabs for 'Query history', 'Query results' (which is active), and 'Table details'. The footer includes a 'Feedback' link, 'English (US)' language setting, and copyright information for Amazon Internet Services Private Ltd.

aws Services Resource Groups

ta15@upgrad.com @ 0679-24... N. Virginia Support

Dashboard Clusters Queries Editor Config Marketplace Advisor Alarms Events

Data objects

Select schema

Select a schema to view data tables

information\_schema

Filter tables and views

applicable\_roles check\_constraints column\_domain\_usage column\_privileges column\_udt\_usage columns constraint\_column\_usage constraint\_table\_usage data\_type\_privileges domain\_constraints

Query editor

redshift-cluster-1 dev awsuser Change connection

Query 1

```
1 select * from stv_inflight;
```

Run Save Clear

Send feedback

Query history Query results Table details

Feedback English (US)

© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

# BASIC SQL OPERATIONS IN REDSHIFT

Demo

The screenshot displays the AWS Redshift console interface. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'ta15@upgrad.com' in the 'N. Virginia' region. The left sidebar contains navigation icons for Dashboard, Clusters, Queries, Editor (highlighted), Config, Marketplace, Advisor, Alarms, and Events.

The main content area is divided into two panels. The left panel, titled 'Data objects', shows a 'Select schema' dropdown set to 'redshift\_demo' and a search bar for 'Filter tables and views'. Below this, a table lists 'redshift\_table'. The right panel, titled 'Query editor', shows a SQL query named 'Query 1' with the following code:

```
1 create schema redshift_demo;
2
3 create table redshift_demo.redshift_table
4 (
5     id int,
6     name char(30),
7     dob date
8 );
```

Below the query editor are buttons for 'Run', 'Save', and 'Clear'. A 'Send feedback' link is also present. The bottom section shows the 'Query results' tab, indicating the query is 'Completed, started on June 25, 2020 at 17:53:26' with an 'ELAPSED TIME: 00 m 16 s'. At the bottom right, there are tabs for 'Execution', 'Data', and 'Visualize'.

# TABLE DESIGN BEST PRACTICES

## Table Compression

Use AZ64 where possible,  
ZSTD/LZO for  
most (VAR)CHAR columns

## Sort Keys

Add sort keys to the columns  
that are frequently filtered  
on

## Distribution Keys

A good distribution key should  
have commonly joined  
columns, with no/less skew

## Define Constraints

Though Redshift does not  
enforce foreign/primary keys,  
optimizer uses those  
constraints to generate more  
efficient query plans

## Use Spectrum

Create external tables for  
infrequently used data and  
query using Spectrum

## Data Types

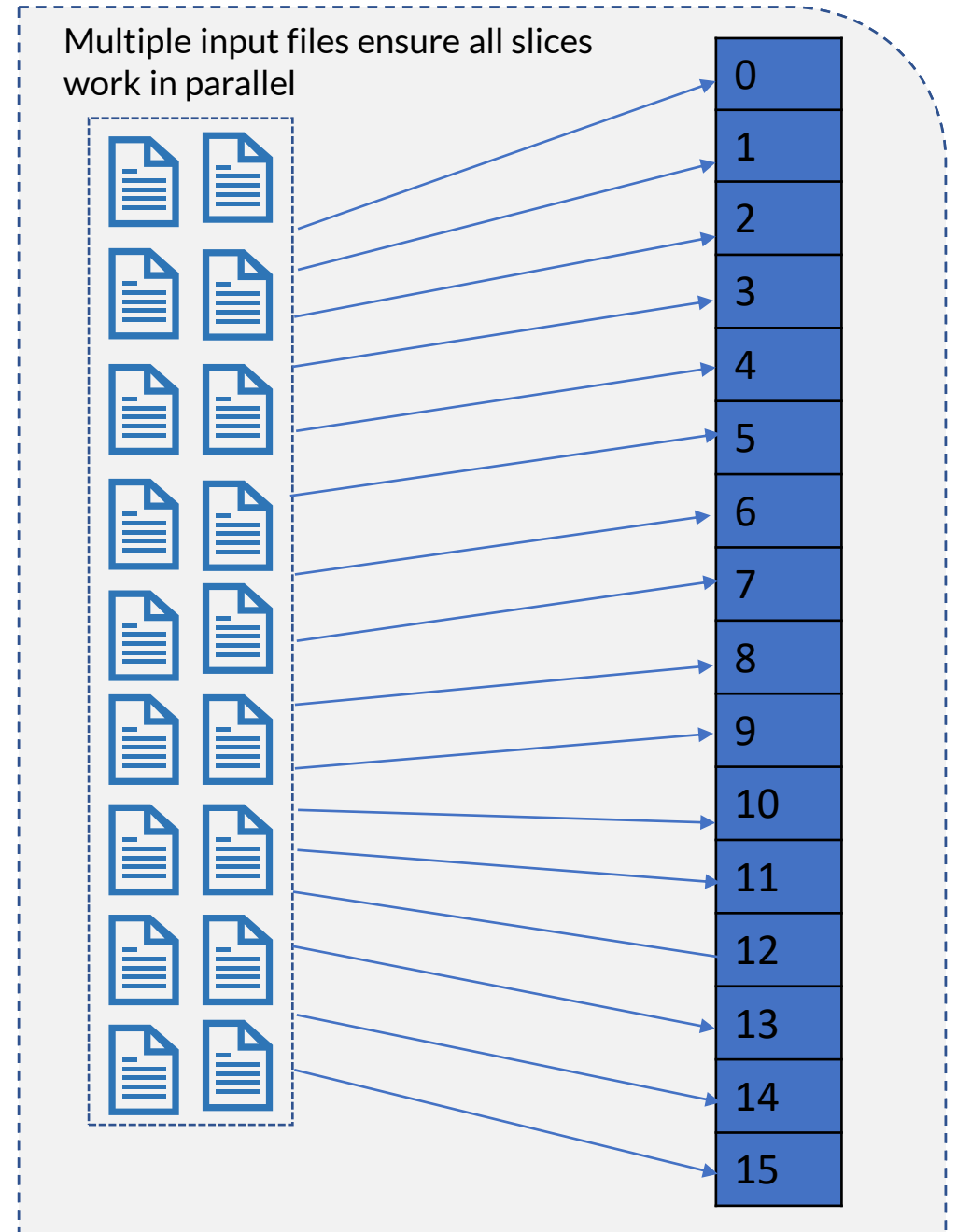
Use appropriate data types,  
such as timestamp for date  
columns instead of  
char/varchar

# LOADING DATA IN REDSHIFT

**COPY** command facilitates loading the data in Redshift from S3

- Redshift is designed to load large amounts of data in parallel
- Write throughput is almost directly proportional to the number of nodes; in other words, the more the number of files, better is the COPY performance
- Recommendation is to use delimited files—1 MB to 1 GB after compression (gzip)
- Number of input files should be a multiple of the number of slices in the Redshift cluster
- Loading data in alternative file formats like Parquet enables significant performance improvement

dc2.8xlarge  
16 slices



# LOADING AND UNLOADING DATA IN REDSHIFT

Demo

The screenshot displays the AWS Redshift console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'ta15@upgrad.com' in 'N. Virginia'. The left sidebar contains icons for Dashboard, Clusters, Queries, Editor (highlighted), Config, Marketplace, Advisor, Alarms, and Events. The main content area is divided into three sections: 'Data objects', 'Query editor', and 'Query results'.

**Data objects:** Shows the 'redshift\_demo' schema selected. A list of tables is displayed: category, date, event, listing, sales, users, and venue.

**Query editor:** Contains a SQL query labeled 'Query 3'.

```
1 copy redshift_demo.users from 's3://redshift-demo-upgrad/ticket/allusers_pipe.txt'
2 iam_role 'arn:aws:iam::067924656089:role/upgrad-redshift-service-role-01'
3 delimiter '|' region 'us-east-1';
4
```

Buttons for 'Run', 'Save', and 'Clear' are visible below the query.

**Query results:** Shows 'Query 778' as 'Completed, started on June 25, 2020 at 18:35:36'. It includes tabs for 'Execution', 'Data', and 'Visualize'.

The bottom of the console features a footer with 'Feedback', 'English (US)', and copyright information for Amazon Internet Services Private Ltd. (2008-2020).

# ANALYSING DATA WITH REDSHIFT

Demo

The screenshot displays the AWS Redshift console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user details (ta15@upgrad.com @ 0679-24..., N. Virginia, Support). The left sidebar contains navigation icons for Dashboard, Clusters, Queries, Editor (highlighted), Config, Marketplace, Advisor, Alarms, and Events.

The main workspace is divided into three sections:

- Data objects:** A panel on the left showing the 'redshift\_demo' schema with tables like category, date, event, listing, sales, users, and venue.
- Query editor:** The central area for writing SQL queries. It shows two queries:
  - Query 1:** Finds the top 10 buyers by quantity.
 

```
-- Find top 10 buyers by quantity.
SELECT firstname, lastname, total_quantity
FROM (SELECT buyerid, sum(qtysold) total_quantity
      FROM redshift_demo.sales
      GROUP BY buyerid
      ORDER BY total_quantity desc limit 10) Q, redshift_demo.users users
WHERE Q.buyerid = userid
ORDER BY Q.total_quantity desc;
```
  - Query 2:** Finds events in the 99.9 percentile in terms of all time gross sales.
 

```
-- Find events in the 99.9 percentile in terms of all time gross sales.
SELECT eventname, total_price
FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc) as percentile
      FROM (SELECT eventid, sum(pricepaid) total_price
            FROM redshift_demo.sales
```
- Query results:** The bottom panel showing the execution status of 'Query 1035'. It indicates the query is 'Completed', started on June 25, 2020 at 18:48:24, with an elapsed time of 00 m 14 s. It also provides options to view the 'Execution', 'Data', or 'Visualize' results.

# UDFs AND STORED PROCEDURES

Demo

## User defined Functions (UDFs)

- Redshift supports custom user-defined functions (scalar) using Python language
- Ability to import custom Python modules makes it a useful feature
- Allows you to create reusable components for tasks such as complex arithmetic calculations
- Cannot use SQL queries within a UDF

Example: Returning the next business day with respect to US Federal Holidays and a M-F work week.

## Stored Procedures

- Redshift supports stored procedures in the PL/pgSQL format, including loops, conditionals, case statements, and IN/OUT/INOUT argument passing
- Allows you better 'lift-and-shift' from traditional data warehouses
- Loops and conditional logic run on the leader node of Amazon Redshift cluster, and any SQL within it is distributed to compute nodes

Example: Running batch SQLs on your redshift cluster

# PERFORMANCE TUNING BASICS

## Redshift Query Lifecycle

### 1. Planning

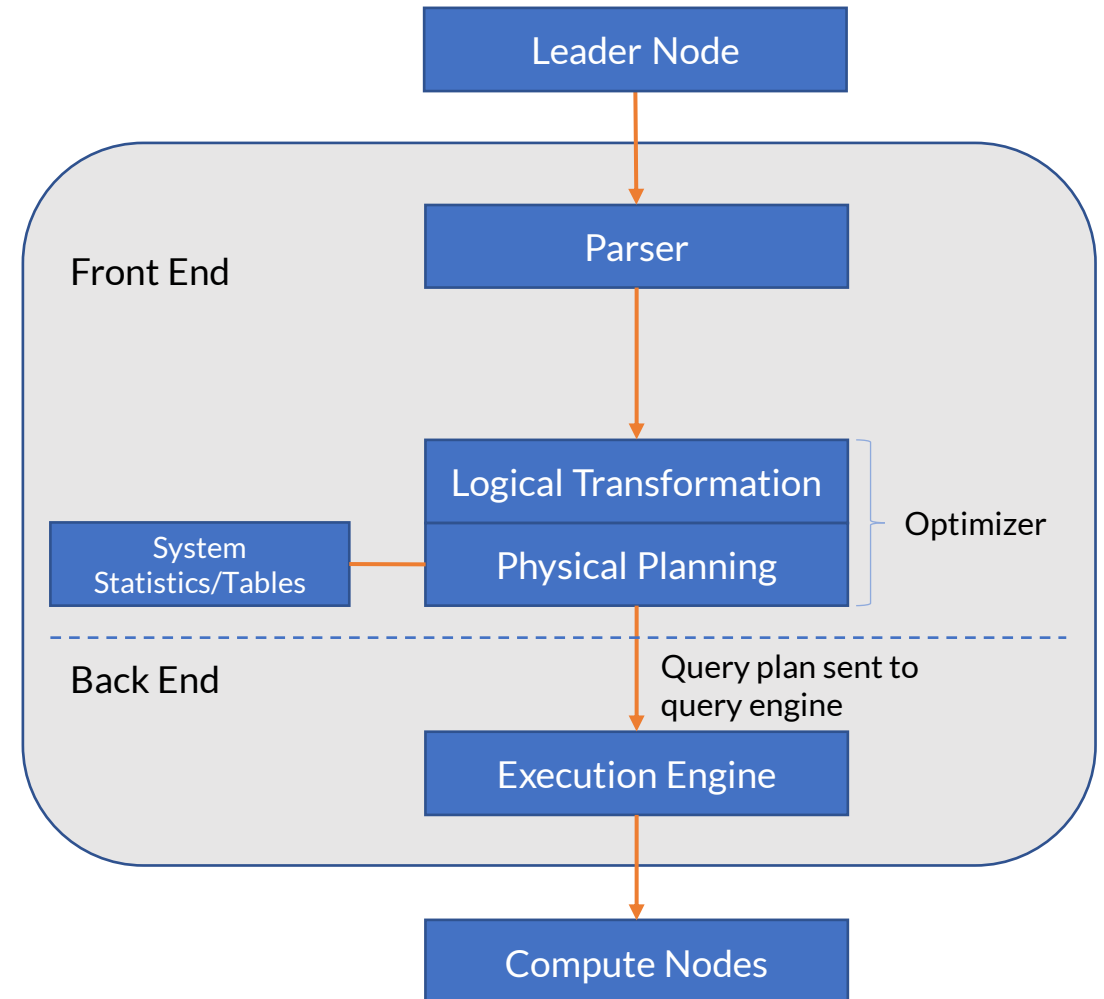
SQL query is parsed and optimized in order to create a query plan

### 2. Compilation

Compile cache is checked for a query plan match; otherwise the query plan is converted to subtasks and subtasks are compiled into C++

### 3. Execution

Compiled code is executed by compute nodes slices in parallel, and results are aggregated by the leader node





# PERFORMANCE TUNING BASICS

Demo

**EXPLAIN** select \* from redshift\_demo.category, redshift\_demo.event where category.catid=event.catid;

## Bad Query Plan

XN Hash Join **DS\_BCAST\_INNER**  
(cost=0.14..**6600286.07** rows=8798  
width=84)  
Hash Cond: ("outer".catid =  
"inner".catid)  
-> XN Seq Scan on event  
(cost=0.00..87.98 rows=8798  
width=35)  
-> XN Hash (cost=0.11..0.11  
rows=11 width=49)  
-> XN Seq Scan on category  
(cost=0.00..0.11 rows=11  
width=49)

## Good Query Plan

XN Hash Join **DS\_DIST\_ALL\_NONE**  
(cost=109.98..**747.87** rows=8798  
width=84)  
Hash Cond: ("outer".catid =  
"inner".catid)  
-> XN Seq Scan on category  
(cost=0.00..0.11 rows=11 width=49)  
-> XN Hash (cost=87.98..87.98  
rows=8798 width=35)  
-> XN Seq Scan on event  
(cost=0.00..87.98 rows=8798  
width=35)

# SESSION SUMMARY

You can connect to a Redshift cluster using any JDBC/ODBC compliant tool

Redshift supports Stored Procedures and UDFs to facilitate easier batch operations

Redshift offers in-built query editor on the console to run queries on the cluster

Redshift integrates with different BI tools seamlessly

Objects (tables, views, functions, etc.) are logically grouped in schemas

EXPLAIN plan defines how the query runs on the cluster nodes and helps in tuning the queries for better performance

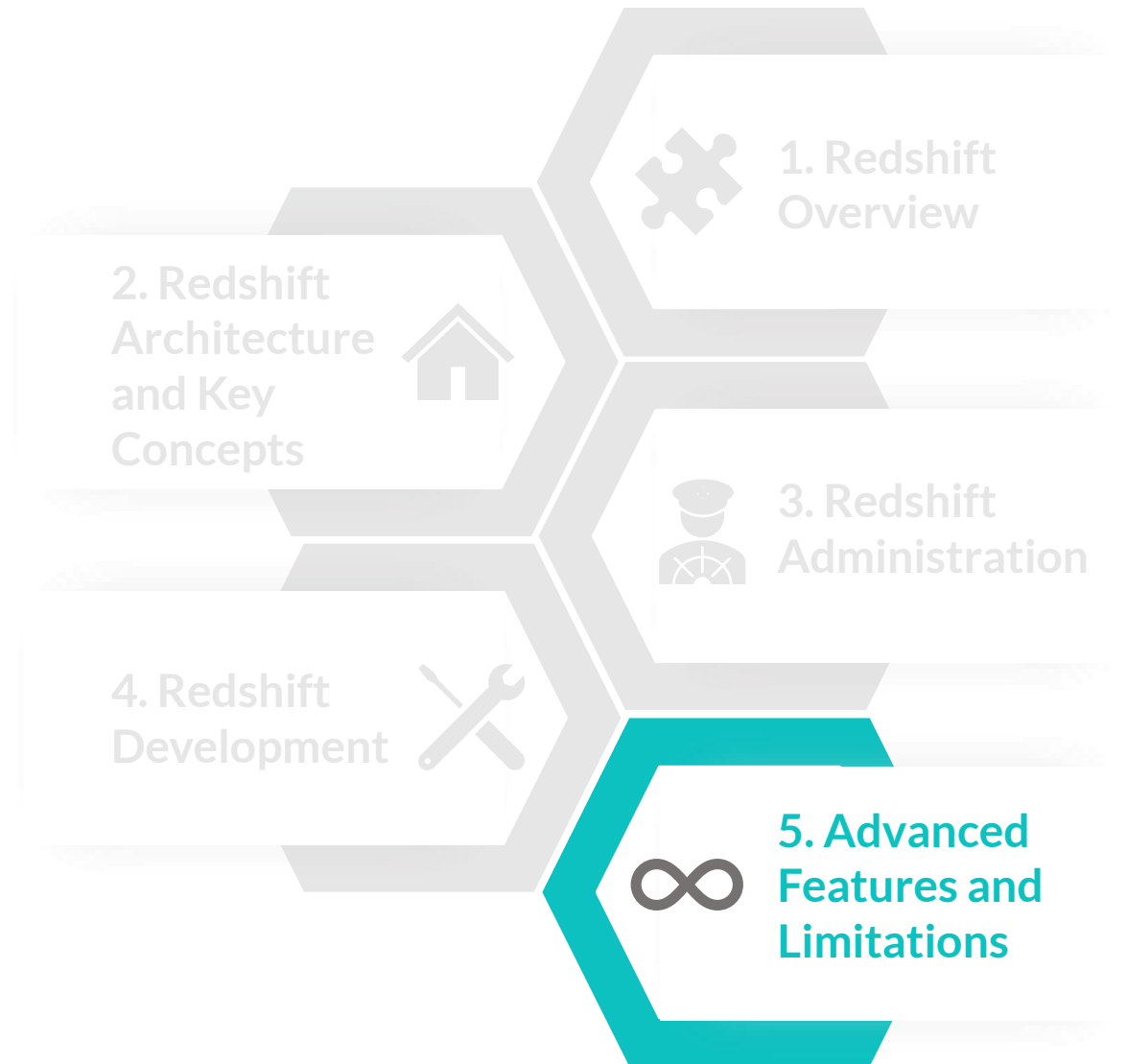
COPY command is used to load data into Redshift table from S3

UNLOAD command helps unload data from Redshift on S3 and supports different file formats, including Parquet

## Session 5: Redshift Advanced Features and Limitations

Upon completion of this session, you will learn:

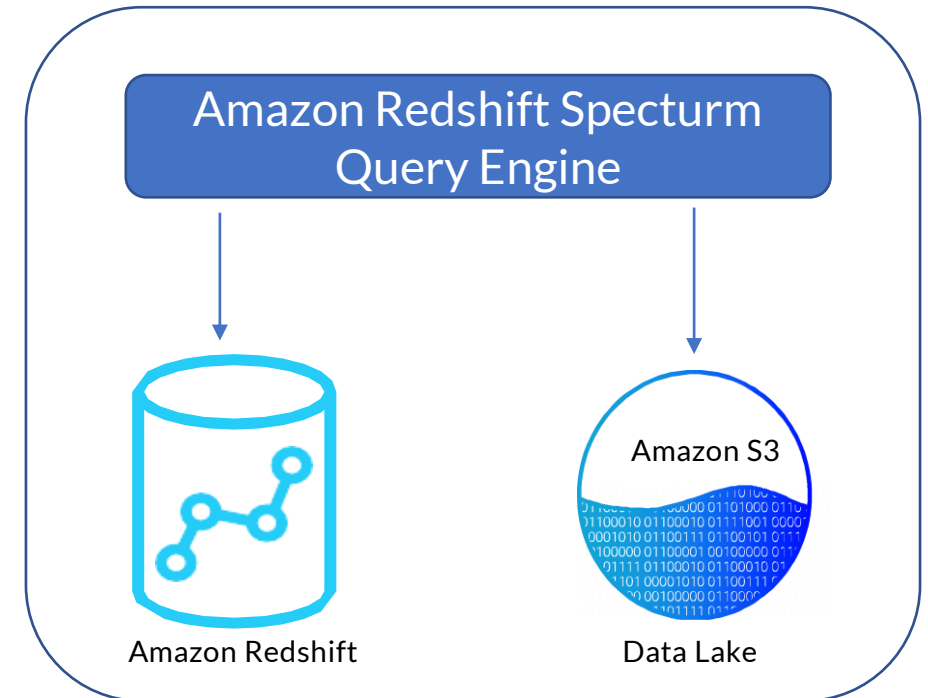
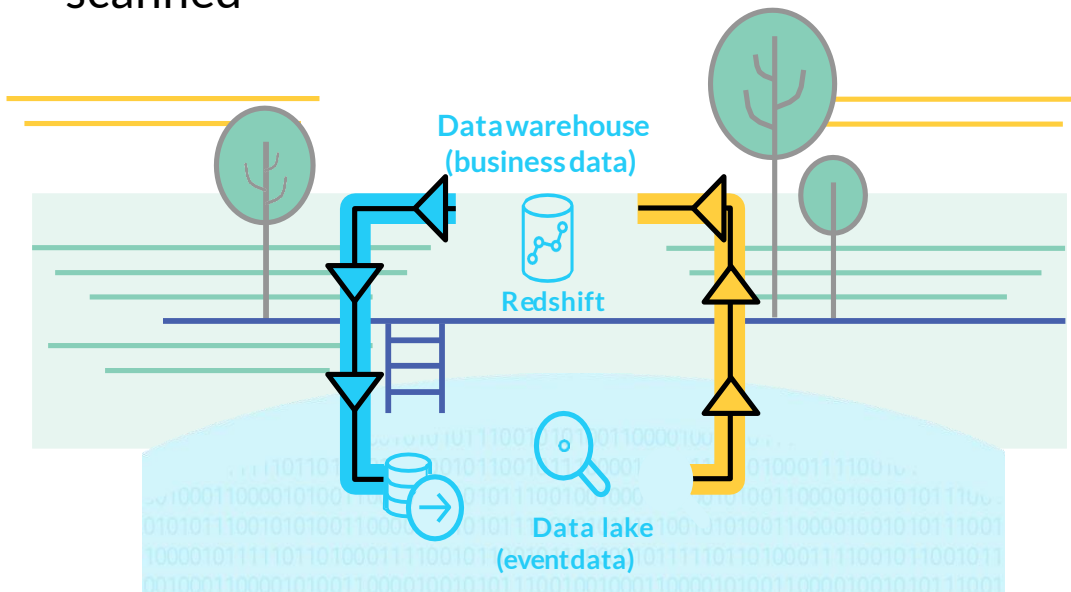
1. About Amazon Redshift Spectrum
2. About Advanced Query Accelerator (AQUA)
3. What are some limitations of Redshift?
4. How do we optimize Redshift costs?



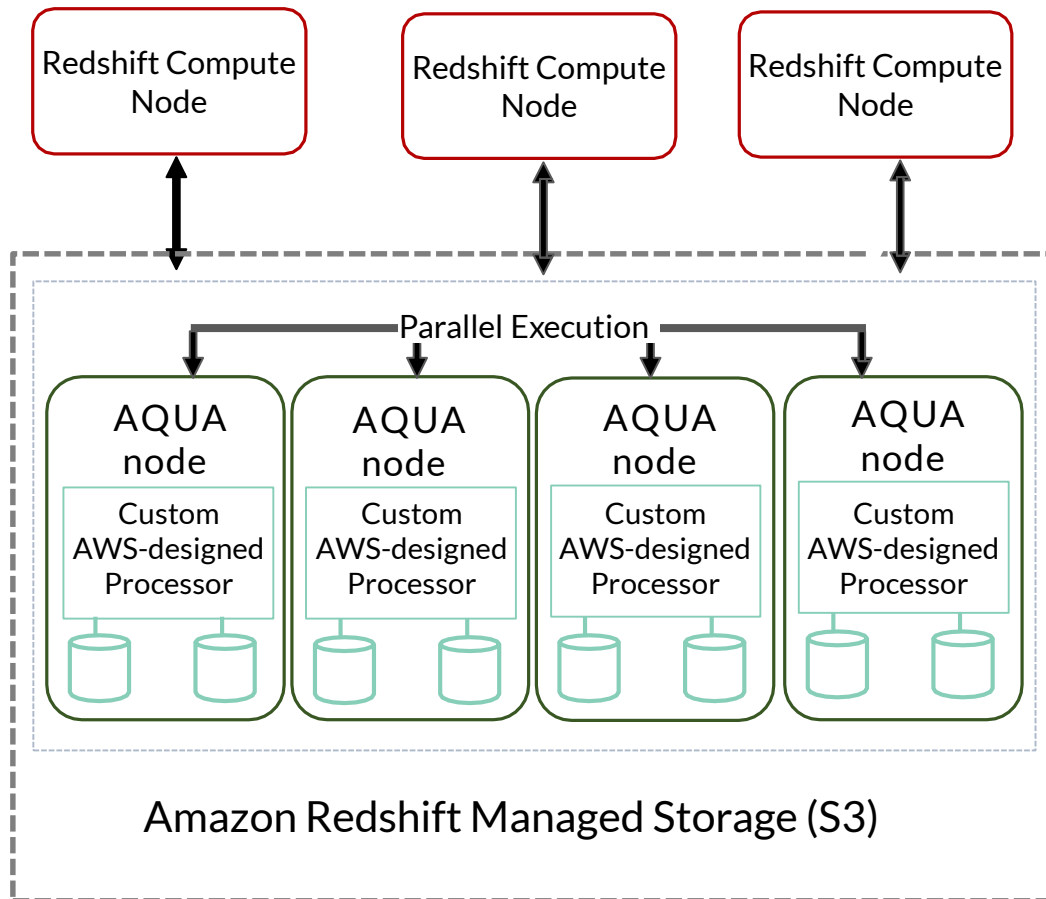
# AMAZON REDSHIFT SPECTRUM

- Redshift Spectrum enables customers to have a **lake house approach**
- Run SQL queries against S3 without loading data into Redshift
- Supports multiple open-source data formats such as CSV, Avro and Parquet
- On-demand pricing; pay per query and data scanned

- Spectrum utilizes a fleet of Amazon Redshift clusters, which are independent of your cluster
- Filtering and aggregation are performed at Spectrum, reducing load on the Redshift cluster



# ADVANCED QUERY ACCELERATOR (AQUA)



New **distributed** and **hardware-accelerated** processing layer

---

With AQUA, Amazon Redshift is up to **10x faster** than any other cloud data warehouse at no extracost

---

AQUA Nodes **with custom AWS-designed analytics processors** to make operations (compression, encryption, filtering, and aggregations) faster than traditional CPUs

---

**Available in Preview with RA3** with no code changes required

---

# REDSHIFT LIMITATIONS

## Learning Curve

Understanding core concepts like sort keys and distribution keys is important, else performance can be a nightmare

## Single Commit Queue

Because of the expense of commit overhead, limit commits by explicitly creating transactions

## Constraints

Redshift does not enforce unique, primary and foreign keys. This could lead to data quality issues

## OLTP

Redshift is not suitable for heavy and frequent write operations

## Loading Data

Currently, you can load data in parallel from S3, DynamoDB, and Amazon EMR.

# OPTIMIZING REDSHIFT COSTS



## Region Selection

Selecting the right AWS region for your cluster is very important as the cost varies per region.

For example, a 1-node ds2.xlarge cluster in US-EAST-1 would cost \$0.85 per hour, but it would cost \$1.36 per hour in SA-EAST-1 (South America)



## Reservations

If you expect the cluster to run for at least a year, it is better to reserve the nodes for a 1-year or 3-year term, which offers significant (20% or 75%) cost savings over the on-demand pricing



## Redshift Spectrum

Offloading infrequently accessed data to S3 and querying it via Redshift Spectrum enables you to use less number of nodes on Redshift cluster, thus saving costs



## Pause & Resume

In order to save costs, you can pause your Redshift clusters, especially in lower environments (development & test stages) when they are not in use.

# OPTIMIZING REDSHIFT COSTS



## Compression

Proper column encoding ensures that data is compressed according to data type. This helps save disk space and costs, besides ensuring performance improvement



## Vacuum

Ensuring that Vacuum is running (either Auto or Manual) helps free up disk space, resulting in cost savings, as less number of nodes will be required



## Snapshots

AWS charges separately for the manual snapshots. This can incur significant costs. An optimal snapshot strategy may help achieve better cost savings



## Rightsizing

Selection of the right size and type of Redshift cluster (memory, CPU, or storage) is important as it can easily lead to over-provisioned resources if not selected properly.



# HELPFUL LINKS AND UTILITIES

<https://github.com/awslabs/amazon-redshift-utils>

<https://github.com/awslabs/amazon-redshift-monitoring>

<https://github.com/awslabs/amazon-redshift-udfs>

## Admin scripts

Collection of utilities for running diagnostics on your cluster

## Admin views

Collection of utilities for managing your cluster, generating schema DDL, and so on

## Analyze Vacuum utility

Utility that can be scheduled to vacuum and analyze the tables within your Amazon Redshift cluster

## Column Encoding utility

Utility that will apply optimal column encoding to an established schema with data already

# SESSION SUMMARY

Redshift Spectrum enables you to query data on S3 and join it with data on Redshift

Reservation of nodes offers 40% to 70% costs savings over on-demand prices

AQUA is a new feature which offers up to 10x performance than other cloud data warehouses at no extra cost

Redshift automated snapshots are not charged, but manual snapshots are charged

Redshift has a single commit queue

Redshift costs varies from region to region

You can reserve Redshift nodes for a 1 year or 3 year term

Most of the maintenance operations in Redshift are automated

# MODULE SUMMARY

01

Amazon Redshift is based on PostgreSQL with additional capabilities to support OLAP operations

02

Redshift offers up to 3x performance than other DW and you can store PBs of data without impacting the performance

03

Redshift does not enforce constraints and does not support traditional indexes; instead it has sort key and distribution keys

04

Using WLM, you can define separate queues for separate workloads and assign portion of cluster memory to each queue

05

Most maintenance activities, like VACUUM and ANALYZE in Redshift are automated

06

Using Redshift Spectrum, you can query the data on S3 and join it with data on Redshift

07

The new RA3 nodes decouple storage from compute and comes with AQUA, which provides up to 10x performance

08

Using COPY command, you can load in parallel into Redshift

09

Redshift offers automated snapshots which can also be scheduled

10

With node reservations, you can save up to 60% costs over on-demand pricing