

Lecture Notes

Amazon Redshift

Setting the Context

Today, information and analytics are irreplaceable for businesses. Almost all significant businesses have built data warehouses for the purposes of detailing and analytics by using data from different sources.

However, the process of building and running a data warehouse—a central repository that stores data coming from one or more data sources—is complicated and expensive. While most information warehousing frameworks are difficult to set up, the other related activities such as planning, acquisition, use and sending of forms can also take months. After you begin your business and set up your data warehouse, you will need to hire skilled and trained database administrators to ensure your inquiries run swiftly and avoid any information loss.

Traditional data warehouses are quite difficult to scale. When the volume of data increases or you want to create analytics and reports and make them available to more clients, you need to choose between tolerating moderate inquiry execution and investing time and effort on a costly overhaul. Many enterprises find it difficult to maintain a healthy relationship with traditional database merchants. The enterprises are frequently compelled to either update their equipment for a managed framework or enter into an extended arrangement cycle with the database merchants with a lapsed term licence. When enterprises reach the scaling limit on one data warehouse engine, they are forced by the same vendor to migrate to another engine but with different SQL semantics.

Comparison Point	On-premise Data warehouse	Cloud Data warehouse
Scalability	As your business changes, you will likely have to purchase new servers or hardware to suit large-scale development on the off chance that you have got an on-premise data warehouse.	A cloud warehouse dispenses with that ultimately, making scaling up (including throughput or capacity) or down much less demanding.
Compatibility	Traditional data warehouses lack the flexibility to work with other analytical engines.	Cloud computing has a feature of plug n play. You can integrate any tool with it for analysis or storing data.
Governance	As data and number of consumers increase, it becomes difficult to	Data security is one of the concerns and cloud computing provides you with complete security on air. You can

	implement governance and control around the data.	set access limitations as per group, role and user.
Variety	Traditional data warehouses are often unable to handle different open data formats, such as Parquet, ORC, and JSON.	Cloud data warehouse is well capable of handling a variety of data formats such as Parquet, CSV, and JSON.











Why Redshift?

Some of the key features that make Amazon Redshift one of the most popular cloud-based data warehousing solutions are summarised in the image below.



Industrial Use Case of Redshift

The aforementioned features are not only feasible but also easy to adapt to; hence, companies can choose to adopt Amazon Redshift.

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

To manage data in bulk, companies are approaching cloud-based warehouses to scale up business on-demand. Redshift is one of the most efficient, fast-paced and robust data warehouse services from AWS.

Redshift - Overview and Architecture

Amazon Redshift is a cloud-based, fully managed data storage service of a petabyte scale. You can begin using Redshift with only a few hundred gigabytes of data and a petabyte or larger storage. This enables you to use your data to derive new insights for your business and customers.

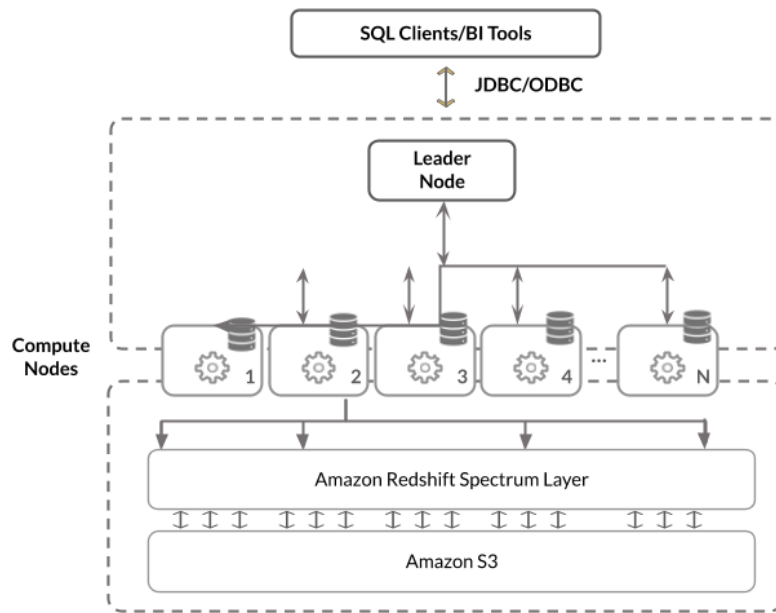
Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyse all the data across your data warehouses and data lakes. It delivers performance that is 10 times faster than other cloud-based data warehouses by using machine learning, massive parallel query execution,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

Amazon Redshift achieves extremely fast query execution by employing the following performance features:

- Massive parallel processing (MPP)
- Online analytical processing (OLAP)
- Columnar storage
- Platform as a service (SaaS)
- Fully managed on the cloud (AWS)

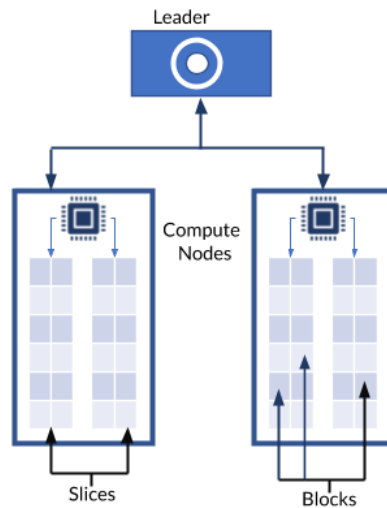
Redshift Architecture

Amazon Redshift is a cloud-based representation of a traditional data warehouse. Redshift requires computing resources to be provisioned and set up in the form of clusters, which contain a collection of one or more nodes. Each node has its own CPU, storage, and RAM. A leader node compiles queries and transfers them to compute nodes, which execute the queries.



On each node, data is stored in chunks, which are called slices. Redshift uses columnar storage, which means that every data block includes values from a single column over a number of rows, rather than a single row of several column values.

1. **Redshift Cluster:** Redshift uses a cluster of nodes as the core component of its infrastructure. Generally, a cluster has one leader node and several compute nodes. If only one compute node exists, the cluster will have no leader node.
2. **Compute Nodes:** Every compute node has its own CPU, storage and memory disk. Client applications are independent of the existence of compute nodes and do not have to deal directly with compute nodes.
3. **Leader Node:** Leader node handles all client communications, manages the coordination between the compute nodes, and query parsing and execution plan development. The leader node generates the execution plan after receiving a query and assigns the compiled code to the compute nodes. Every compute node is allocated a portion of the data. The Leader node performs the final aggregation of the results.



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 has a portion of memory and disk space allocated to the node.
- 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 Data Storage

Amazon Redshift organises the data in columns. Unlike row-based systems, which are ideal for processing transactional data, column-based systems are generally used for analytics and data warehousing. In such systems, queries often involve aggregates performed over large data sets, and these systems allow only the columns involved in the queries to be processed. Hence, column-based systems require far fewer I/Os (disk I/Os), thereby improving query performance significantly.

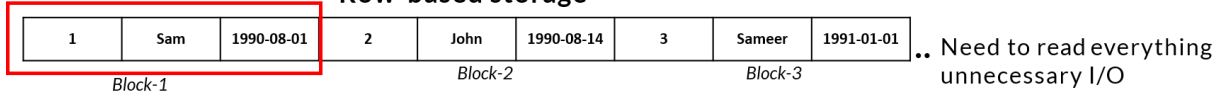
The table below is an illustration of columnar data search.

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

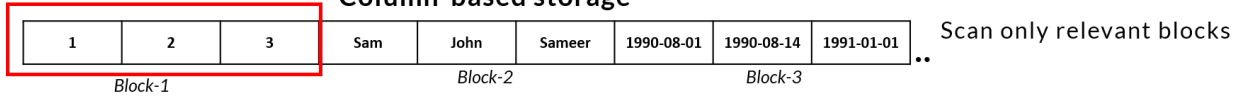
id	loc	dob
1	Sam	1990-08-01
2	John	1990-08-14
3	Sameer	1991-01-01

Redshift executes the above query and the below images show how redshift internally processes data.

Row-based storage



Column-based storage

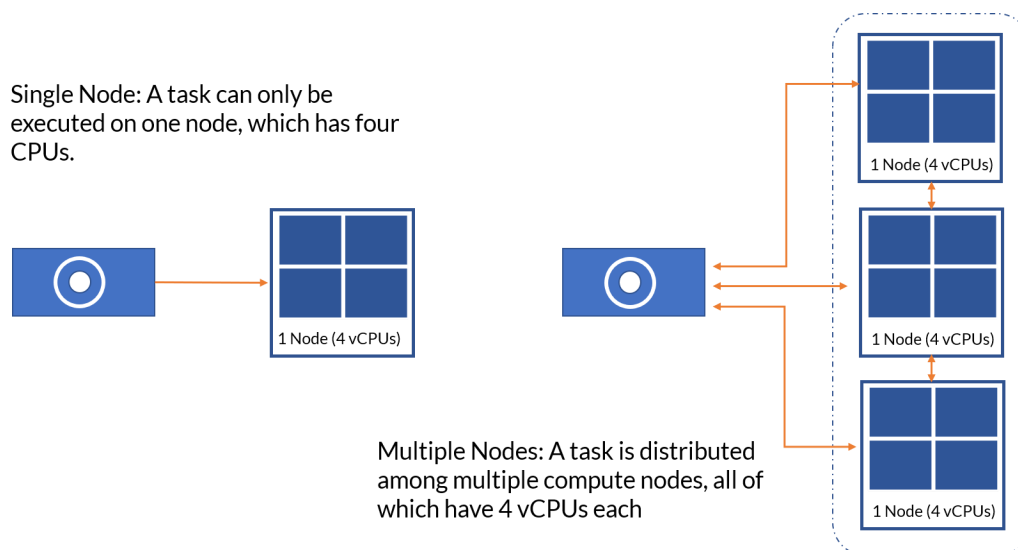


Data Compression

Data compression eliminates additional storage demands, thereby minimising the I/O space, which increases the efficiency of the database. The compressed data is read from the memory first and then uncompressed after query execution. The loading of a small volume of data into memory enables Amazon Redshift to allocate more resources to data processing. Since columnar storage holds identical data sequentially, Amazon Redshift implements adaptive compression encoding directly related to columnar data forms. The easiest way to create table column data compression is by enabling Amazon Redshift to implement optimised compression encoding while loading the table with the results.

Massive Parallel Processing (MPP)

Amazon Redshift automatically distributes data and inquiry stack over all nodes. It helps you to easily include centres into your data warehouse and enables you to perform quick query execution as your data warehouse grows.



Massive parallel processing (MPP) allows for quick execution of the most complicated queries. Multiple compute nodes perform the data processing, leading to the aggregation of the final response, with each node core running the same assembled database segments on the portions of the actual results.

Amazon Redshift distributes the table rows among the compute nodes so that the data can be stored in parallel. You can automate the delivery of the data by choosing the appropriate delivery key for each table to manage the workload and reduce the transfer of data from one node to another.

SORT key

When loading data into the table, the data will be sorted by one or more columns designated as sort keys. SORT key is a keyword to specify a single-column sort key after a column name, or you can specify one or more columns as the table sort key columns using the syntax SORTKEY (column name, [...]). With this syntax, only compound sort keys will be developed.

If no sort keys are defined, then the list will not be sorted. You can use up to 400 SORTKEY columns per row.

As Redshift writes blocks for each object, it stores some metadata. An important part of the data stored in each block is the minimum and maximum values that produce maps of the region. Zone maps will dramatically speed up queries by allowing Redshift to skip entire blocks while searching for information.

Types of Sort Key

Single Column	Compound	Interleaved																																																															
SORTKEY (Date)	SORTKEY COMPOUND (Date, Region)	SORTKEY INTERLEAVED (Date, Region, Country)																																																															
<table> <tr> <th>Date</th><th>Region</th><th>Country</th></tr> <tr> <td>2-JUNE-2020</td><td>OCEANIA</td><td>NEW ZEALAND</td></tr> <tr> <td>2-JUNE-2020</td><td>ASIA</td><td>SINGAPORE</td></tr> <tr> <td>2-JUNE-2020</td><td>AFRICA</td><td>ZAMBIA</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>HONG KONG</td></tr> <tr> <td>3-JUNE-2020</td><td>EUROPE</td><td>GERMANY</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>KOREA</td></tr> </table>	Date	Region	Country	2-JUNE-2020	OCEANIA	NEW ZEALAND	2-JUNE-2020	ASIA	SINGAPORE	2-JUNE-2020	AFRICA	ZAMBIA	3-JUNE-2020	ASIA	HONG KONG	3-JUNE-2020	EUROPE	GERMANY	3-JUNE-2020	ASIA	KOREA	<table> <tr> <th>Date</th><th>Region</th><th>Country</th></tr> <tr> <td>2-JUNE-2020</td><td>AFRICA</td><td>ZAMBIA</td></tr> <tr> <td>2-JUNE-2020</td><td>ASIA</td><td>SINGAPORE</td></tr> <tr> <td>2-JUNE-2020</td><td>OCEANIA</td><td>NEW ZEALAND</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>HONG KONG</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>KOREA</td></tr> <tr> <td>3-JUNE-2020</td><td>EUROPE</td><td>GERMANY</td></tr> </table>	Date	Region	Country	2-JUNE-2020	AFRICA	ZAMBIA	2-JUNE-2020	ASIA	SINGAPORE	2-JUNE-2020	OCEANIA	NEW ZEALAND	3-JUNE-2020	ASIA	HONG KONG	3-JUNE-2020	ASIA	KOREA	3-JUNE-2020	EUROPE	GERMANY	<table> <tr> <th>Date</th><th>Region</th><th>Country</th></tr> <tr> <td>2-JUNE-2020</td><td>OCEANIA</td><td>NEW ZEALAND</td></tr> <tr> <td>2-JUNE-2020</td><td>ASIA</td><td>SINGAPORE</td></tr> <tr> <td>2-JUNE-2020</td><td>AFRICA</td><td>ZAMBIA</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>HONG KONG</td></tr> <tr> <td>3-JUNE-2020</td><td>EUROPE</td><td>GERMANY</td></tr> <tr> <td>3-JUNE-2020</td><td>ASIA</td><td>KOREA</td></tr> </table>	Date	Region	Country	2-JUNE-2020	OCEANIA	NEW ZEALAND	2-JUNE-2020	ASIA	SINGAPORE	2-JUNE-2020	AFRICA	ZAMBIA	3-JUNE-2020	ASIA	HONG KONG	3-JUNE-2020	EUROPE	GERMANY	3-JUNE-2020	ASIA	KOREA
Date	Region	Country																																																															
2-JUNE-2020	OCEANIA	NEW ZEALAND																																																															
2-JUNE-2020	ASIA	SINGAPORE																																																															
2-JUNE-2020	AFRICA	ZAMBIA																																																															
3-JUNE-2020	ASIA	HONG KONG																																																															
3-JUNE-2020	EUROPE	GERMANY																																																															
3-JUNE-2020	ASIA	KOREA																																																															
Date	Region	Country																																																															
2-JUNE-2020	AFRICA	ZAMBIA																																																															
2-JUNE-2020	ASIA	SINGAPORE																																																															
2-JUNE-2020	OCEANIA	NEW ZEALAND																																																															
3-JUNE-2020	ASIA	HONG KONG																																																															
3-JUNE-2020	ASIA	KOREA																																																															
3-JUNE-2020	EUROPE	GERMANY																																																															
Date	Region	Country																																																															
2-JUNE-2020	OCEANIA	NEW ZEALAND																																																															
2-JUNE-2020	ASIA	SINGAPORE																																																															
2-JUNE-2020	AFRICA	ZAMBIA																																																															
3-JUNE-2020	ASIA	HONG KONG																																																															
3-JUNE-2020	EUROPE	GERMANY																																																															
3-JUNE-2020	ASIA	KOREA																																																															
<ul style="list-style-type: none"> Best for queries that use the 1st column as primary filter Can speed up joins and group by statements Quickest to VACUUM 	<ul style="list-style-type: none"> Table sorted by order as given in sort key Best for queries that use the 1st column as primary filter, then others Slower to VACUUM 	<ul style="list-style-type: none"> Equal weight given to each column Best for queries that use different filter columns Slowest to VACUUM 																																																															

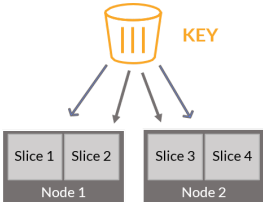
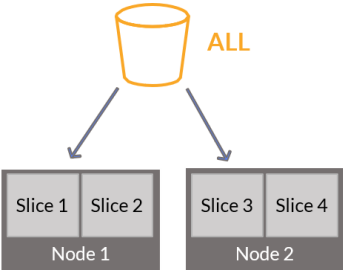
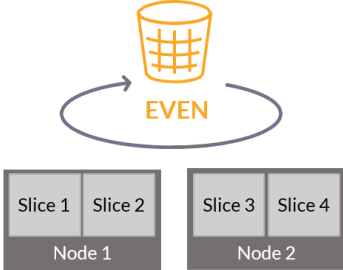
ZONE map

While creating a table, you can specify one or more columns as the sort key. Amazon Redshift holds the data on the disk in a sorted order. Whether or not the data is organised has a significant impact on the disk I/O, column compression and query performance.

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	<ul style="list-style-type: none"> 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

Data distribution is a plan that defines how data will be stored on the compute nodes.

Distribution types available in Redshift:

Illustration	Type
	<p>KEY</p> <p>Value is hashed, and the same value goes to the same location (slice). Having the DistKey as the foreign key for both the rows, you can ensure that the joined data in each row is still on the same node, thereby minimising data redistribution.</p>
	<p>ALL</p> <p>The full table data goes to the first slice of each node. ALL will distribute the data among all the nodes. This is good for reading performance, as data is always local to the computed node, but the write and maintenance costs are high.</p>
	<p>EVEN</p> <p>This distribution type follows the round-robin rule. It distributes data uniformly across networks, which means that each node receives an equal share of data. This type is appropriate when the table does not need to be attached to other tables.</p>

DISTRIBUTION key

The **DIST KEY** distributes **rows** among the slices. If two tables have the same **DIST KEY**, then all the rows in *both tables* with the same value for the **DIST KEY** column will be located on the same slice. Note that dates and timestamps are not eligible for the **DIST KEY** because they are rarely used in a **JOIN**. Unique identifiers, such as `product_id`, would be a better **DIST KEY**.

Redshift Administration

The Redshift cluster consists of one or more nodes. Each cluster contains just one leader and one or more compute nodes. If you have a cluster with only one node in it, this node assumes two roles - leader and compute.

When the leader node receives a query, it first parses the query and turns it into an optimised execution plan. These steps will then be compiled into the code and distributed to the compute nodes. When the

compute nodes return intermediate result sets, the leader merges these together and replies to the client application.

Node Types

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

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

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

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

Amazon Redshift turbocharges query performance with machine learning-based automatic optimisations.

VACUUM

- VACUUM removes rows that are marked as deleted and globally sorts tables.
- For the majority of the 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.
- Use VACUUM BOOST at off-peak times (blocks deletes), which is as quick as 'Deep Copy'.

ANALYZE

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

Workload Management allows for the separation of different query workloads, given below.

Goals

- Prioritise important queries
- Throttle/abort less important queries

- Control concurrent number of executing queries
- Divide cluster memory
- Set query timeouts to abort long-running queries

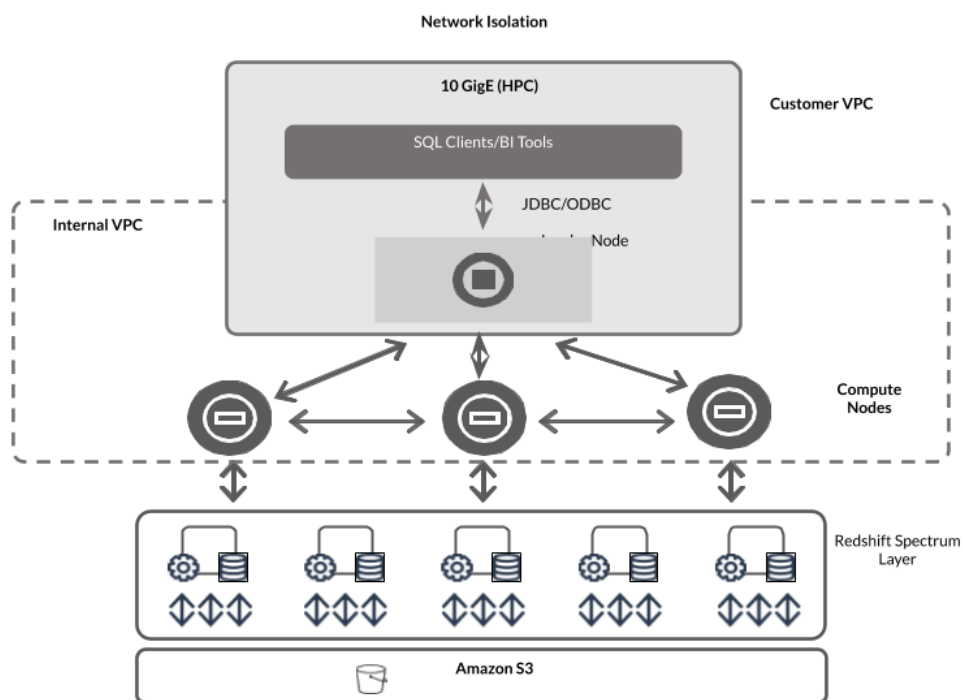
Queues

- Assign a percentage of cluster memory
- SQL queries executed in the 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)

Amazon Redshift turbocharges query performance with the following machine learning-based automatic optimisations.



Redshift Development

AWS has customised PostgreSQL to form Redshift. Therefore, inside Redshift, the PostgreSQL rules are to be followed. While querying in the Redshift cluster, you have to follow all key aspects that are present in PostgreSQL. Here, you will also find some restrictions, for example, the 'use' keyword is not used to set schema for a session. Since Redshift is designed only for OLAP functions, it allows a limited number of functions.

Here are some key points to remember:

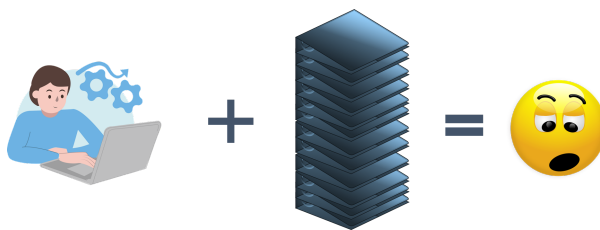
- While creating a table, the schema name should be present before the table name else it is created in a public schema.
- While inserting data into a table, you must provide the correct schema name, else you will get an error. Or, if you forget to add the schema name, then that table may get created as a public schema and data may get inserted into that table.

You have executed the basic SQL query and created the table with data insertion. But for an optimised query and better performance, you should have the best practice of writing optimal SQL queries in Redshift.

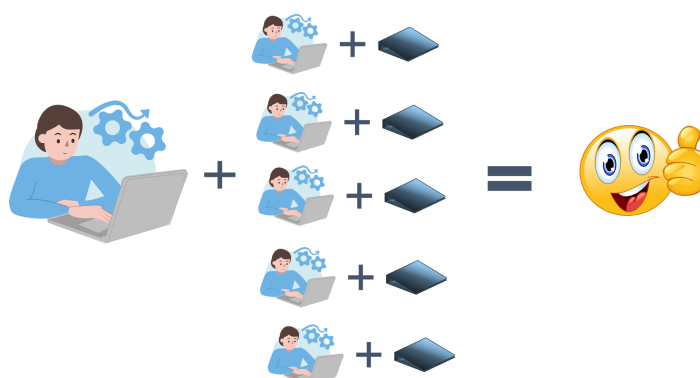
The **COPY** command is always directly proportional to the number of **Nodes**. Redshift automatically decompresses the file while loading using the **COPY** command.

Let's understand the COPY command with an example.

Mohan has 1,600 files, which he has to review in a few hours. This puts the whole pressure on one person. This would yield less efficient results and also decrease the quality of work done.

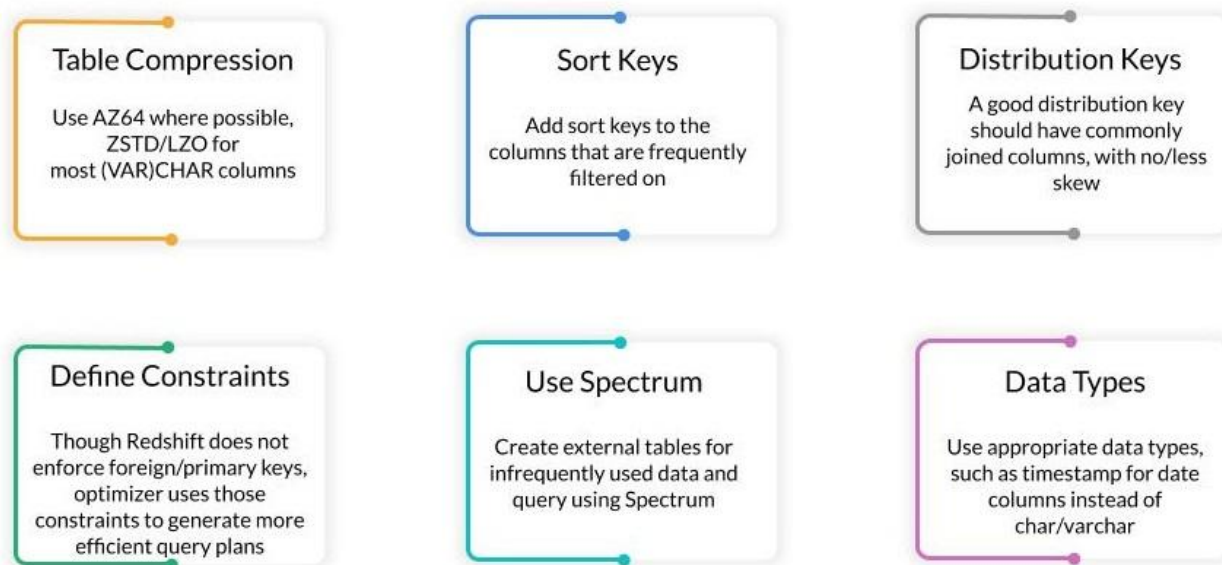


If Mohan hires 16 people or takes help from others and distributes the work among them, each person will get to review 100 files. This increases work efficiency and also helps achieve on-time delivery.



The example given above is just an illustration to help you understand the concept of the **COPY** command. Let's understand the practical approach of the **COPY** command.

Some best practices that you have to follow



User-Defined Functions (UDFs)

- Redshift supports custom user-defined functions (scalar) using Python language or a single SELECT statement.
- 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.

Stored Procedures

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

You can see the reference for UDF below:

```
CREATE OR REPLACE FUNCTION f_next_business_day(dt DATE)
RETURN date
STABLE
AS $$
import pandas
from pandas.tseries.offsets import CustomBusinessDay
from pandas.tseries.holiday import USFederalHolidayCalendar
bday_us = CustomBusinessDay(calendar= USFederalHolidayCalendar())
return dt + bday_us

$$ LANGUAGE plpythonu;
```

In a similar manner, SQL queries need tuning for better performance.

The examples below illustrate the operation performed over a query and show how to get a query plan by executing the **EXPLAIN** command.

Bad Query Plan

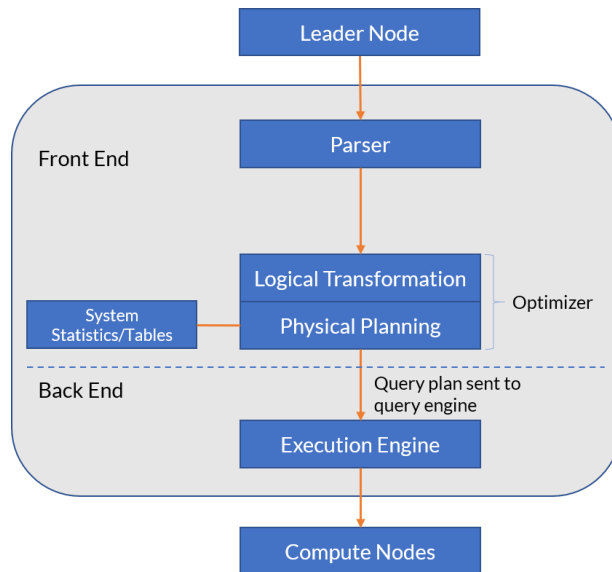
```
-> XN Hash Join DS_BCAST_INNER (cost=112.50..3272334142.59 rows=170771 width=84)
    Hash Cond: ("outer".venueid = "inner".venueid)
-> XN Hash Join DS_BCAST_INNER (cost=109.98..3167290276.71 rows=172456 width=47)
    Hash Cond: ("outer".eventid = "inner".eventid)
-> XN Merge Join DS_DIST_NONE (cost=0.00..6286.47 rows=172456 width=30)
    Merge Cond: ("outer".listid = "inner".listid)
-> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497 width=14)
-> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=24)
```

Good Query Plan

```
-> XN Hash Join DS_DIST_ALL_NONE (cost=112.50..14142.59 rows=170771 width=84)
    Hash Cond: ("outer".venueid = "inner".venueid)
-> XN Hash Join DS_DIST_ALL_NONE (cost=109.98..10276.71 rows=172456 width=47)
    Hash Cond: ("outer".eventid = "inner".eventid)
-> XN Merge Join DS_DIST_NONE (cost=0.00..6286.47 rows=172456 width=30)
    Merge Cond: ("outer".listid = "inner".listid)
-> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497 width=14)
-> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=24)
```

After changing the dimension tables to use DISTSTYLE ALL, the query plan for the same query shows DS_DIST_ALL_NONE in place of DS_BCAST_INNER. Also, there is a dramatic change in the relative cost for the join steps.

Below image illustrating how queries get planned and executed.



The query planning and execution workflow follow these steps:

1. The leader node receives the query and parses the SQL.
2. The parser produces an initial query tree that is a logical representation of the original query. Amazon Redshift then inputs this query tree into the query optimiser.
3. The optimiser evaluates and, if necessary, rewrites the query to maximise its efficiency. This process sometimes creates multiple related queries to replace a single one.
4. The optimiser generates a query plan (or several, if the previous step resulted in multiple queries) for execution with the best performance. The query plan specifies execution options such as join types, join order, aggregation options, and data distribution requirements.
5. You can use the [EXPLAIN](#) command to view the query plan. A query plan is a fundamental tool for analysing and tuning complex queries. For more information, see [Query plan](#).
6. The execution engine translates the query plan into *steps*, *segments* and *streams*.
7. The compute node slices execute the query segments in parallel. As part of this process, Amazon Redshift takes advantage of the optimised network communication, memory, and disk management to pass intermediate results from one query plan step to the next; this also helps to speed up query execution.

After changing the dimension tables to use `DISTSTYLE ALL`, the query plan for the same query shows `DS_DIST_ALL_NONE` in place of `DS_BCAST_INNER`. Also, there is a dramatic change in the relative cost for the join steps.

[Supported data types](#)

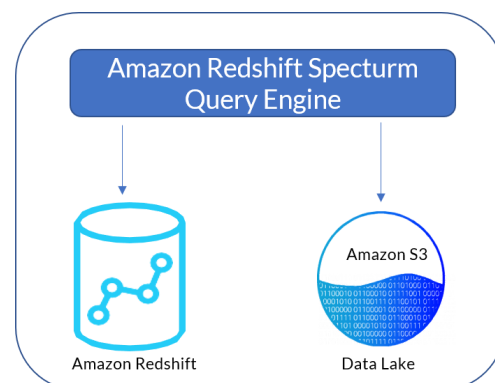
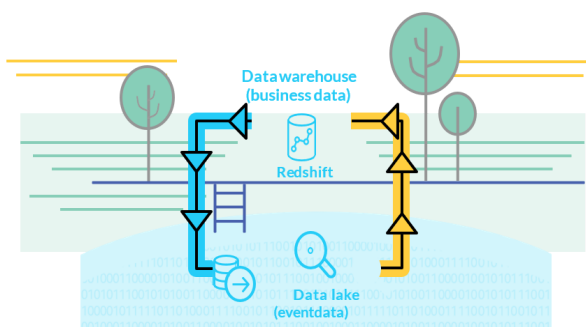
Advance Feature and Development

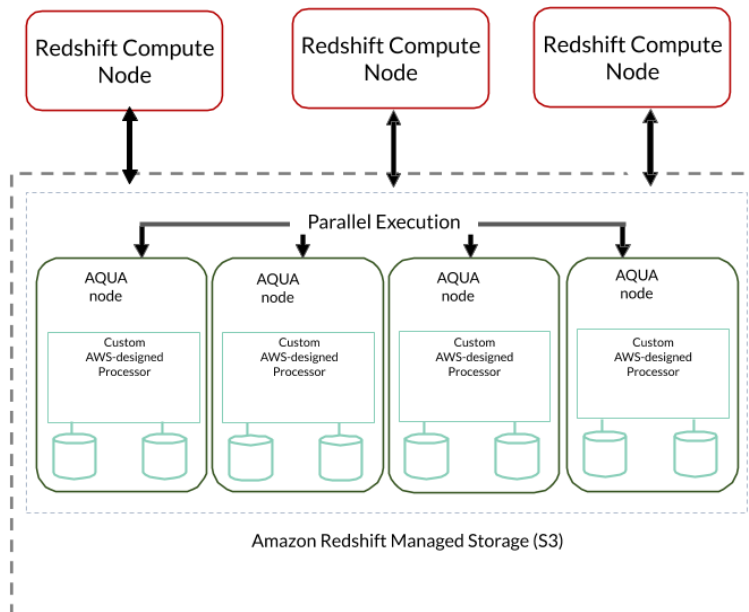
Redshift Spectrum

An analyst can query SQL on stored data from Amazon S3 buckets using Redshift Spectrum. It helps save time, as it eliminates the need to transfer data from a storage facility to a database. Redshift Spectrum also expands the spectrum of a specific application, as it applies to broad volumes, ranging from unstructured S3 data lakes to the current Redshift data storage nodes of a customer.

Redshift Spectrum breaks the user query into filtered subsets that run simultaneously. These requests are spread across thousands of AWS-managed nodes to maintain high query speed and ensure consistency in performance. Redshift Spectrum can be scaled up to run a query across more than an exabyte of data. Once the data in S3 is aggregated, it is returned to a local Redshift cluster for final processing.

- Redshift Spectrum enables customers to have a lake house approach.
- It runs SQL queries against S3 without loading data into Redshift.
- It supports multiple open-source data formats such as CSV, Avro and Parquet.
- It offers on-demand pricing as well as pay per query and data scanned.
- The spectrum utilises a fleet of Amazon Redshift clusters, which are independent of your cluster.
- Filtering and aggregation are performed at Spectrum, reducing the load on the Redshift cluster.

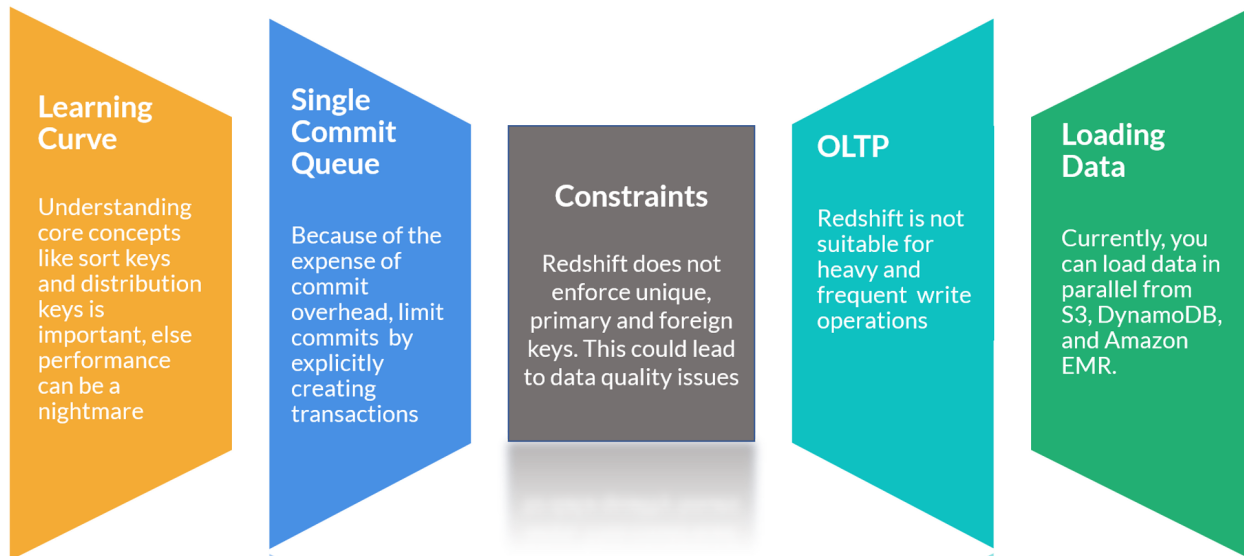




- With AQUA, Amazon Redshift is up to 10x faster than any other cloud data warehouse at no extra cost.
- AQUA Nodes with custom AWS-designed analytics processors enable operations (compression, encryption, filtering, and aggregations) faster than traditional CPUs.

Limitations of Amazon Redshift

1. Despite being a completely managed service, Amazon Redshift needs some user intervention for vacuuming.
2. The pricing of Redshift includes computing and storage. Separating these two features is not possible. However, alternatives such as Snowflake can enable this.
3. Data load and transfer involving non-AWS services are complex in Redshift. Using a platform such as Hevo Data can significantly improve this experience.
4. Redshift is not custom-made for real-time operations but is suited for batch operations. A platform such as Hevo Data can help you with real-time operations.
5. Redshift scaling is not completely seamless, but it includes a small window of downtime where the cluster is not available for querying. While this downtime is in the range of minutes for newer-generation nodes that use elastic scaling, it is in the range of hours for older-generation nodes.



Redshift Restrictions

- **Deduplication:** In Redshift, it is not possible to implement the authenticity of the data added. Therefore, if you have a distributed network and want to write data to Redshift, then you have to either deal with the uniqueness of the application layer or utilise any code deduplication process.
- **Parallel upload:** If the data is in Amazon S3, Amazon DynamoDB or Amazon EMR, then Redshift will load it using the Massive Parallel Processing, which is quite easy. However, parallel loading is not supported for all other sources. You will have to use JDBC inputs or other scripts to load data into Redshift.
- **Requires a good understanding of Sort and Dist keys:** Sort keys and distribution keys determine how data is stored and indexed across all nodes in Redshift. Therefore, you need to have a good knowledge of these principles, and you need to position them correctly in your tables for optimal results. Note that there should be only one delivery key for a table, and it cannot be modified afterwards; this ensures that you plan carefully to predict potential workloads before settling on the Dist key.
- **Live app database:** Although Redshift is quick enough to run queries on a large volume of files or run monitoring and analytics, it is not good enough for live web applications. So, you will need to pull Redshift data into a caching layer or a vanilla Postgres instance to serve it to web applications.

Optimisation Techniques

Region Selection	<p>Selecting the correct AWS region for your cluster is important, as the cost varies by region.</p> <p>For example, a 1-node DS2.xlarge cluster in US-EAST-1 would cost \$0.85 per hour, but in SA-EAST-1 (South America), it would cost \$1.36 per hour.</p> <p>For this module, use the 'US-EAST-1' region only.</p>
Reservations	<p>If you expect the cluster to run for at least a year, it is better to reserve the nodes for a one-year or three-year term, which offers significant (20% or 75%) cost savings over on-demand pricing.</p>
Redshift Spectrum	<p>Offloading infrequently accessed data to S3 and querying it via Redshift Spectrum enables you to use fewer nodes on a Redshift cluster, thus saving costs.</p>
Pause and Resume	<p>To save costs, you can pause your Redshift clusters, especially in lower environments (development and test stages) when they are not in use.</p>
Compression	<p>Proper column encoding ensures that data is compressed according to the data type. This helps save disk space and costs, besides improving performance.</p>
Vacuum	<p>Ensuring that Vacuum is running (either Auto or Manual) helps free up disk space, resulting in cost savings, as fewer nodes will be required.</p>
Snapshots	<p>AWS charges separately for manual snapshots. This can incur significant costs. On the other hand, an optimal snapshot strategy may help achieve better cost savings.</p>
Rightsizing	<p>Selecting the right size and type of a Redshift cluster (memory, CPU or storage) is important, as it can easily lead to over-provisioned resources if not selected properly.</p>

Some points to remember:

- Redshift Spectrum enables you to query data on S3 and join it with data on Redshift.
- Redshift has a single commit queue.
- Most of the maintenance operations in Redshift are automated.
- Redshift automated snapshots are not charged, but manual snapshots are charged.
- Reservation of nodes offers 40-70% cost savings over on-demand prices.
- Redshift nodes can be reserved for a one-year or threeterm.

Below are some helpful links:

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

Below are some useful utilities:

Admin scripts	Collection of utilities for running diagnostics on your cluster
Admin views	Collection of utilities for managing your cluster, generating schema DDL, etc.
Analyse Vacuum utility	A utility that can be scheduled to vacuum to analyse the tables within your Amazon Redshift cluster
Column Encoding utility	A utility that will apply optimal column encoding to an established schema with data already

This module has been summarised in a single image, as shown below.

- 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 decouples 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

Disclaimer: *All content and material on the upGrad website is copyrighted material, belonging to either upGrad or its bona fide contributors, and is purely for the dissemination of education. You are permitted to access, print, and download extracts from this site purely for your own education only and on the following basis:*

- *You can download this document from the website for self-use only.*
- *Any copies of this document, in part or full, saved to disc or to any other storage medium, may be used for subsequent, self-viewing purposes or to print an individual extract or copy for non-commercial personal use only.*
- *Any further dissemination, distribution, reproduction, and copying of the content of the document herein, or the uploading thereof on other websites, or use of the content for any other commercial/unauthorized purposes in any way that could infringe the intellectual property rights of upGrad or its contributors is strictly prohibited.*
- *No graphics, images, or photographs from any accompanying text in this document will be used separately for unauthorized purposes.*
- *No material in this document will be modified, adapted, or altered in any way.*
- *No part of this document or upGrad content may be reproduced or stored on any other website or included in any public or private electronic retrieval system or service without upGrad's prior written permission.*
- *Any rights not expressly granted in these terms are reserved.*