TABLE SIZE OPTIMIZATION FOR SMALL DATASETS IN AMAZON REDSHIFT

White Paper

AXTRIA
INGENIOUS INSIGHTS

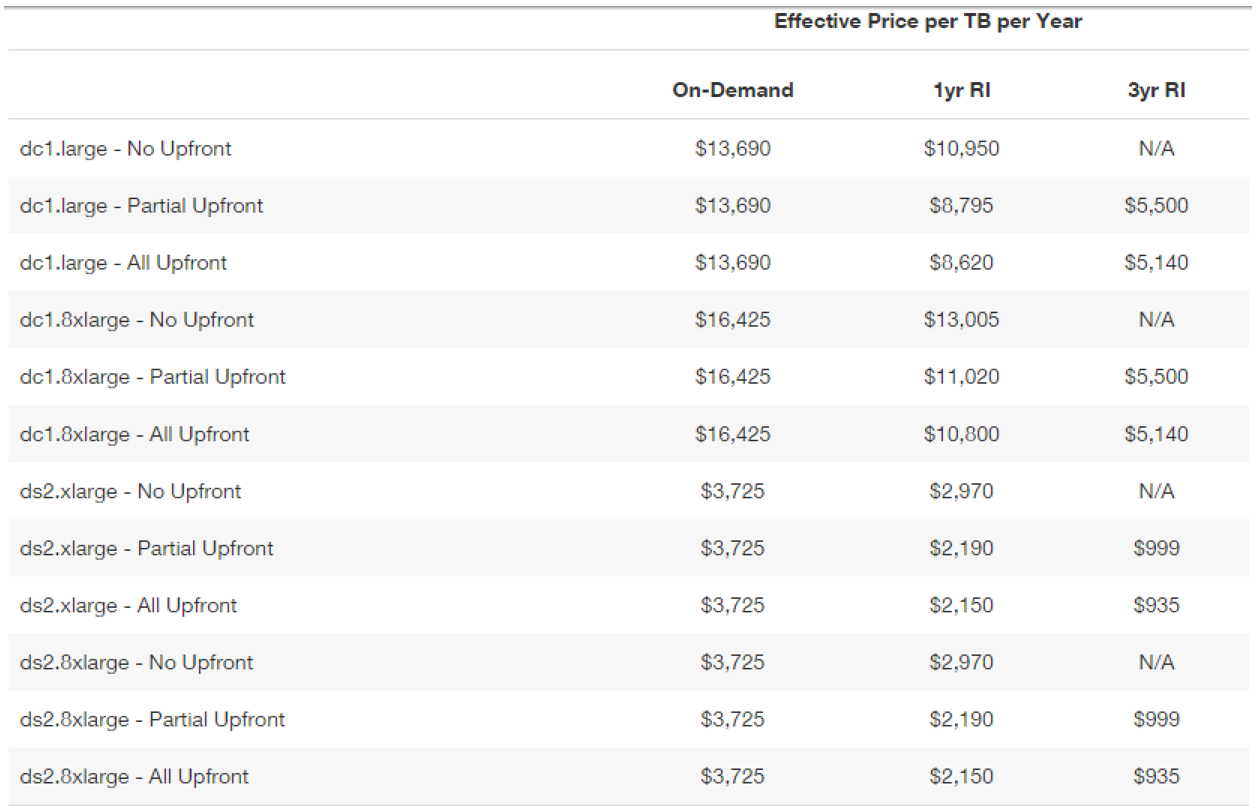
Table of Contents

1. INTRODUCTION

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse. One drawback with Redshift is that it has a high degree of space redundancy when dealing with small datasets (with column sizes less than 1 MB) along with large datasets like xref, lookup, and configuration tables etc. With space being a very important component of the Redshift pricing structure, optimizing the space becomes very important.

The figure below gives an idea of the Redshift pricing. In this whitepaper, we will identify the various factors that affect the space occupancy of a Redshift table, the common pitfalls and will also define methods for reducing the space requirements.

Table 1.1 Redshift pricing



	Effective Price per TB per Year		
	On-Demand	1yr RI	3yr RI
dc1.large - No Upfront	$13,690	$10,950	N/A
dc1.large - Partial Upfront	$13,690	$8,795	$5,500
dc1.large - All Upfront	$13,690	$8,620	$5,140
dc1.8xlarge - No Upfront	$16,425	$13,005	N/A
dc1.8xlarge - Partial Upfront	$16,425	$11,020	$5,500
dc1.8xlarge - All Upfront	$16,425	$10,800	$5,140
ds2.xlarge - No Upfront	$3,725	$2,970	N/A
ds2.xlarge - Partial Upfront	$3,725	$2,190	$999
ds2.xlarge - All Upfront	$3,725	$2,150	$935
ds2.8xlarge - No Upfront	$3,725	$2,970	N/A
ds2.8xlarge - Partial Upfront	$3,725	$2,190	$999
ds2.8xlarge - All Upfront	$3,725	$2,150	$935

2. FACTORS AFFECTING THE TABLE SIZE

We can divide the factors that influence table size into two categories.

1. Static factors: These are the factors that form an essence of the Redshift architecture, largely not modifiable.
    - Type and Number of Nodes
    - Columnar storage and Number of Blocks
2. Dynamic factors: These are factors that a table designer can influence directly. These include
    - Data Distribution Style
    - Number of Rows and Columns

As a part of this presentation, we will look at each of these factors in more detail.

2.1. Static Factors

2.1.1. Type and Number of nodes

An Amazon Redshift data warehouse is a collection of computing resources called NODES. Each node has its own operating system, dedicated memory, and dedicated disk storage (slices). One node is the LEADER NODE; it manages the distribution of data and query processing tasks to the COMPUTE NODES.

The disk storage for a compute node is divided into a number of SLICES. The number of slices per node depends on the node type. For example, each DS1.XL compute node has two slices, and each DS1.8XL compute node has 16 slices. The number of slices for a node is static and cannot be influenced by a Redshift table designer.

Figure below illustrates the storage mapping for different types of Nodes and also their number of slices.

Table 2.1 Dense Storage Node Types

NODE SIZE	VCPU	ECU	RAM(GiB)	SLICES PER NODE	STORAGE PER NODE	NODE RANGE	TOTAL CAPACITY
ds1.xlarge	2	4.4	15	2	2 TB HDD	1-32	64 TB
ds1.8xlarge	16	35	120	16	16 TB HDD	2-128	2 PB
ds2.xlarge	4	13	31	2	2 TB HDD	1-32	64 TB
ds2.8xlarge	36	119	244	16	16 TB HDD	2-128	2 PB

Table 2.2 Dense Compute Node Types

NODE SIZE	VCPU	ECU	RAM(GiB)	SLICES PER NODE	STORAGE PER NODE	NODE RANGE	TOTAL CAPACITY
dc1.xlarge	2	7	15	2	160 GB SSD	1-32	5.12 TB
ds1.8xlarge	32	104	244	32	2.56 TB SSD	2-128	326 TB

The number of slices affects the table size in a big way once a data distribution style is chosen. This is because the distribution style directly determines how many slices will be used for data storage and what data will be stored in each slice. We will look at this in more detail in the case study section.

Columnar storage and number of blocks

The smallest unit of storage in Redshift is called a block. The size of a Redshift block is 1MB. The size of a Redshift block cannot be modified. Each data block stores values of a single column for multiple rows.

If the column data size is less than 1MB then storage for an entire record will take less than one block, resulting in an inefficient use of disk space. Also, two columns do not share the same block. This results in a lot of space wastage

To see the block usage of a Redshift table we can query either the metadata view svv_diskusage or the table stv_blocklist. Each row in these tables represents a block of storage.

As an illustration, a table with 1 record and 99 columns will occupy 99 Blocks of storage translating into 99 MB of space usage. This is the minimum space requirement for a 99 column table.

2.2. Dynamic Factors

2.2.1. Data distribution style

When a query is executed, the query optimizer redistributes the rows to the compute nodes as needed to perform any joins and aggregations. The goal in selecting a table distribution style is to minimize the impact of the redistribution step by physically locating the data where it needs to be before the query is executed. Below are the various distribution styles that can be used.

The distribution style is the biggest factor affecting the table size.

Even distribution

The leader node distributes the rows across the slices in a round-robin fashion, regardless of the values in any particular column. This is the default distribution style for Redshift

This distribution style is the worst for small datasets. An illustration for this is shown in the case study.

Even distribution is the default distribution applied to a table when no distribution style is chosen. This is the most common pitfall in most table designs and results in a lot of space wastage.

Key distribution

The rows are distributed according to the values in one column. The leader node will attempt to place matching values on the same node slice.

This distribution style is also not good when the key column has a high cardinality, only suitable for columns with low cardinality.

All distribution

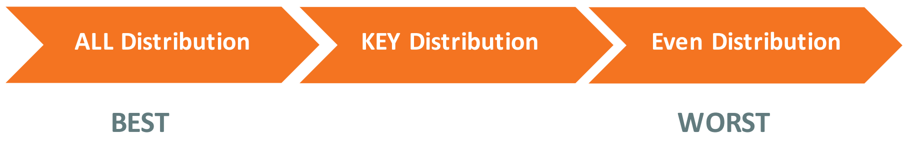
A copy of the entire table is distributed to every node. Where EVEN distribution or KEY distribution place only a portion of a table's rows on each node, ALL distribution ensures that every row is collocated for every join that the table participates in.

This distribution style is the best for small datasets. ***Illustration in the case study in section 3.3.***

2.2.2. Number of rows and columns

The number of columns directly affects the number of blocks used. Each column is stored in a separate memory block. A block is never shared between columns.

If a table has 100 columns then it will occupy a minimum storage of 100 MB (1MB per column). Depending on the distribution style the size of the table increases.



It's surprising but true that the number of rows in case of a small data set does not directly affect the table size, but coupled with the Distribution style it affects in a big way. This will be dealt in detail in the case study section of the presentation.

3. CASE STUDY

As a part of the case study, we will analyze the effect of the factors listed in section 2 on the size of the table DMN_CALN

Number of Nodes and slices: 4 Dc1.8xlarge dense compute nodes. Each node has 32 slices with 128 slices in total.
Number of Rows: 9133
Number of columns: 99
Raw File size: 4.5 MB (each column not more than 1 MB)

Based on the number of columns DMN_CALN will occupy a minimum of 99 MB of space. The distribution style will further inflate the table size.

The way to optimize the table size is to choose the right distribution style and make sure that the table size inflates the least.

3.1. Even Distribution

Case

Number of slices used = 128

Number of columns=99

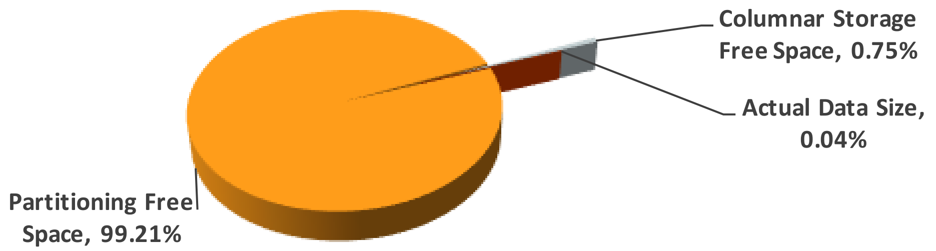
Number of Blocks occupied = 12672 (99*128)

Table size = 12672 MB which is 12GB

Explanation

When an even distribution style is applied to a table, equal number of rows are allocated to all the node slices. In this case, 72/73 rows will be allocated to each slice (72*128 = 9133 - the row count). In each slice 99 blocks will be occupied-As this is the nature of a columnar database this leads to a table size of around 12 GB (99 MB*128=12.6 GB).

As an effect, there is a massive size inflation - from 4.5 MB (text file size) to 12 GB.



3.2. Key Distribution

Case 1

Unique key values= 5

Number of slices used = 5

Number of columns=99

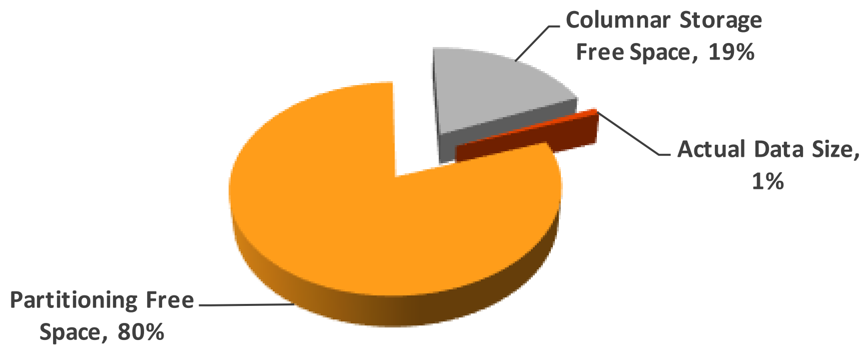
Number of Blocks occupied = 495 (99*5)

Table size = 495 MB which is close to .5 GB

Explanation for Case 1

When a key distribution style is applied to a table, Redshift stores similar key values on the same node to promote co-location and reduce redistribution of data in case of joins. In this case, 5 slices (1 in each node) will be used, each having 99 blocks of data (corresponding to the column count) - As this is the nature of a columnar database. A total of 495 (5*99) blocks will be occupied.

This inflates the size of the table data from 4.5 MB (text file size) to .5 GB



Case 2

Unique key values= 4

Number of slices used = 4

Number of columns=99

Number of Blocks occupied = 396 (99*4)

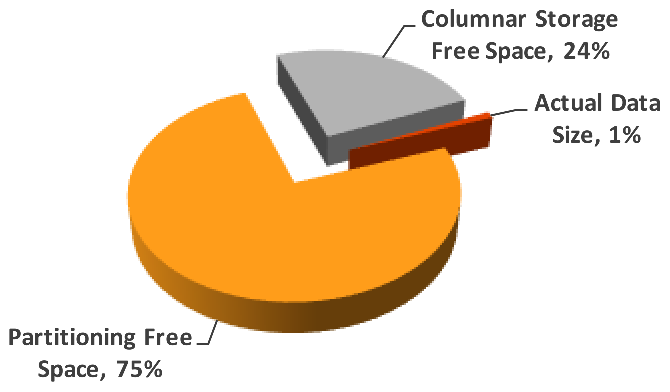
Table size = 396 MB

Explanation for Case 2

When a key distribution style is applied to a table, Redshift stores similar key values on the same node to promote co-location and reduce redistribution of data in case of joins. In this case, 4 slices (1 in each node) will be used, each having 99 blocks of data (corresponding to the column count) - As this is the nature of a columnar database. A total of 396 (4*99) blocks will be occupied

This inflates the size of the table data from 4.5 MB (text file size) to 396 MB

From Case 1 and 2 we see that for the same amount of data when the cardinality is reduced the table size also reduces.



3.3. All Distribution

Case

Number of slices used = 4

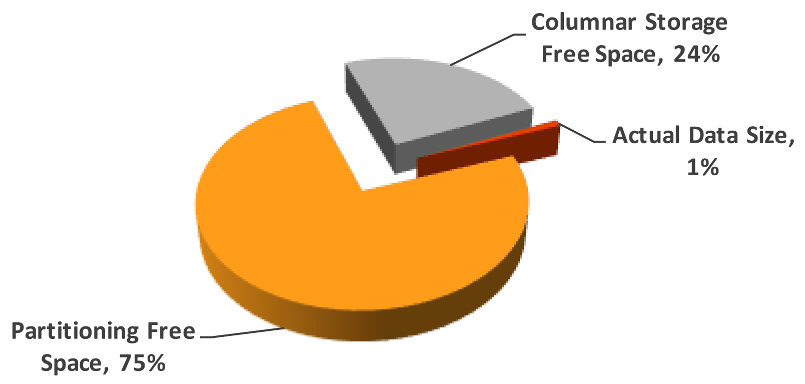
Number of columns=99

Number of Blocks occupied = 396 (99*4)

Rows per Slice= 9133(all nodes have all the 9133 rows)

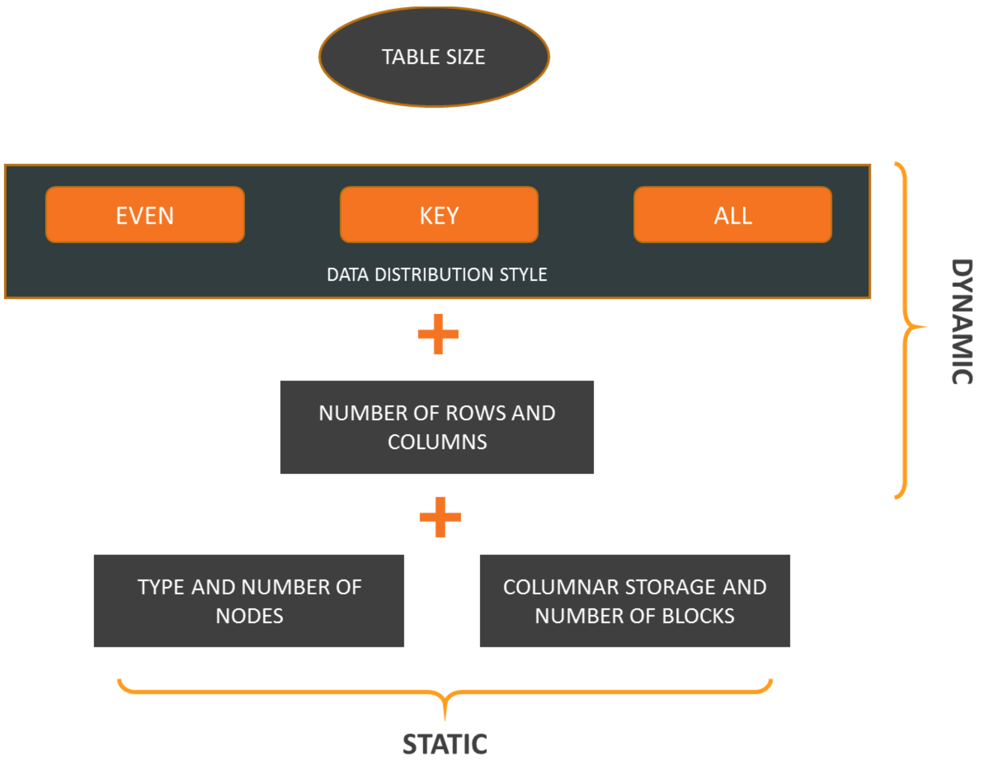
Explanation

When the ALL distribution style is applied to a table, Redshift stores all the values on all the nodes. In this case, 4 slices (1 in each node) will be used, each having 99 blocks of data (corresponding to the column count) - As this is the nature of a columnar database. A total of 396 (4*99) blocks will be occupied.



4. SUMMARY AND RECOMMENDATIONS

1. Amazon Redshift does not allow the remaining space in a block to be reutilized; this inherently leads to a large amount of space wastage.
2. Redshift will inflate the data-size of small databases no matter what is done. The space optimization only deals with the reduction in space inflation.
3. The space utilization is an additive effect of the static and dynamic factors. *The figure below best represents this :*



4. As a rule of thumb, one should never apply even distribution to small datasets.
5. Another rule of thumb - if time is a concern in table design; apply all distribution to small datasets (as shown in Case 3). Another advantage of this approach is that normally small datasets are reference datasets, when a query is run on multiple nodes than each node has a copy of the reference data. This helps to limit the redistribution of data on to multiple nodes.

Recommendation

Create an extra column with a single value in it and apply key partitioning on that column. The data will occupy the least amount of space.

Unique key values= 1

Number of slices used = 1

Number of columns=100

Number of Blocks occupied = 100 (100*1)

Table size = 100 MB

Explanation

In this case, only 1 slice (data will only be on 1 node) will be used for data storage with 100 blocks of data (corresponding to the column count) - As this is the nature of a columnar database. A total of 100 (100*1) blocks will be occupied

The inflation of the table size very limited and the final size is just 100 MB.

The only downside to this approach is redistribution of data (as it resides on only one node) when a query runs on multiple nodes.

About the Authors:

Ravi Shankar - Director
ravi.shankar@axtria.com

Ravi has over 11 years of experience working as a techno functional consultant, with significant experience in data warehousing and Big Data engagements including scoping study, requirements gathering, data architecture design, development, and implementation for end-to-end IT solutions.

Swarup Dessai - Engagement Manager
Swarup.dessai@axtria.com

Swarup has over seven years of experience in Data Warehousing and Business Intelligence.