

Chapter 12: Snowflake Cache and Query Performance.

One of the most important concepts is the cache to improve the speed of our queries in Snowflake and optimize costs. In this chapter, we will study the existing ones (Metadata, Query Result, and Warehouse cache) and some tips to improve Snowflake's performance.

1. [Metadata Cache](#)
2. [Query Result Cache](#)
3. [Warehouse Cache](#)
4. [Complete example of the Snowflake caches](#)
5. [How to improve Snowflake performance](#)
6. [Typical exam questions on Snowflake caches](#)

METADATA CACHE

Metadata caching is maintained in Global Service Layer, and it contains Objects Information & Statistics. As we mentioned in the chapter about micro-partitions, Snowflake automatically stores different types of metadata to improve the compiling time and query optimization. This metadata information lasts for 64 days. This cache will help us perform operations like MIN, MAX, COUNT... In these cases, Snowflake will NOT use the warehouses, so we don't spend computing credits to do that.

Remember that we also studied that you cannot copy the same file into Snowflake using the COPY INTO command unless you specify the option FORCE=TRUE during 64 days? Now you understand who is in charge of that.

Just as an example, we can have a 30TB table with 931 million rows and get the COUNT of all the values from the table in microseconds, without performing anything:

Status	Success	SQL Text	1 SELECT COUNT(*) FROM "ANALYTICS"."PUBLIC".MYTABLE
User			
Warehouse			
Start Time	4:42:35 PM		
End Time	4:42:36 PM		
Total Duration	238ms		
Scanned Bytes	0		
Rows	0		
Query ID		Query Result	
Session ID		Results	
		row#	COUNT(*)
		1	931640656

Metadata cache example.

QUERY RESULT CACHE

Have you ever tried to execute the same query twice, and the second time is way faster? Let's take a look at this example. We are going to run the same query twice, the second time after 7 hours.

```
SELECT * FROM ANALYTICS.PUBLIC.MYTABLE ORDER BY DATE
DESC;
```

If we go to the History tab in the SnowFlake UI, we can see the results of our queries, apart from filtering by different params, like the user in this case. As we can see at the bottom of this picture, SnowFlake spent the first time 14.3 seconds to perform the query, whereas the second time, it spent 47ms.

History

Hide Filters View SQL Abort...

Display queries that meet all of the following criteria:

User is

☐ Include client-generated statements

☐ Include queries executed by user tasks

Status	Query ID	SQL Text	User	Warehouse	Clust...	Total Duration	Size	Bytes Scanned	Rows
Success		SELECT * FROM 'ANALYT...				47ms			
Success		SELECT * FROM 'ANALYT...			1	14.3s	X-Small	251.4MB	9.0M

Snowflake History tab.

How is this possible? This is because of the Query Result Cache. This cache stores the results of our queries for 24 hours, so as long as we perform the same query and the data hasn't changed in the Storage layer, it will return the same result without using the warehouse. So again, we don't consume compute credits to perform the same query all the times we want.

You cannot see the results from other people in the History tab, but Snowflake stores the result so that if different people (with the same role) perform the same query, they will also use this cache.

You can disable the Query Result cache with the following command:

```
ALTER SESSION SET USE_CACHED_RESULT = FALSE
```

WAREHOUSE CACHE

Every warehouse has attached SSD storage. So, while the data warehouse runs, the table fetched in the query will remain there. When the warehouse is suspended, the information will be lost. Let's see an example. In this case, we are going to perform a similar query, just changing the columns to show.

1st query, we fetch all the columns from *MyTable*:

```
SELECT * FROM ANALYTICS.PUBLIC.MYTABLE ORDER BY DATE DESC;
```

2nd query, we fetch the columns "Date" and "Score" from *MyTable*:

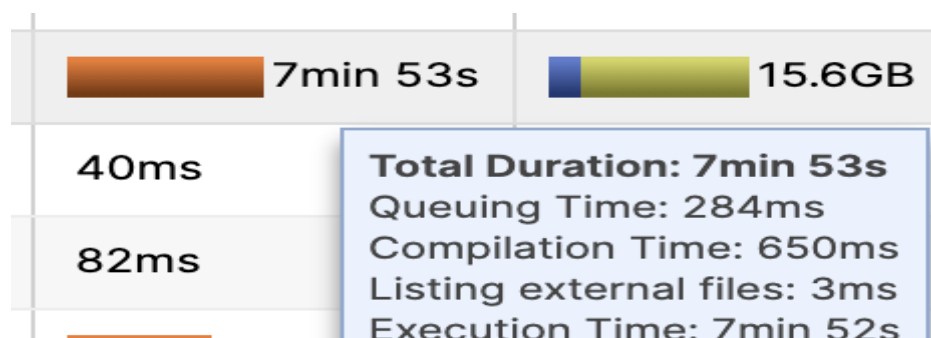
```
SELECT DATE, SCORE FROM ANALYTICS.PUBLIC.MYTABLE ORDER BY DATE DESC;
```

Why isn't the second query re-used from the Query Result Cache? Because it's *NOT* the same query, we are selecting fewer columns. But let's take a look at the History tab:

Status	Query ID	Cluster Number	Size	Total Duration	Rows	Bytes Scanned
✓	2	1	Medium	59.3s	933.8M	6.2GB
✓	1	1	Medium	7min 53s	933.8M	15.6GB

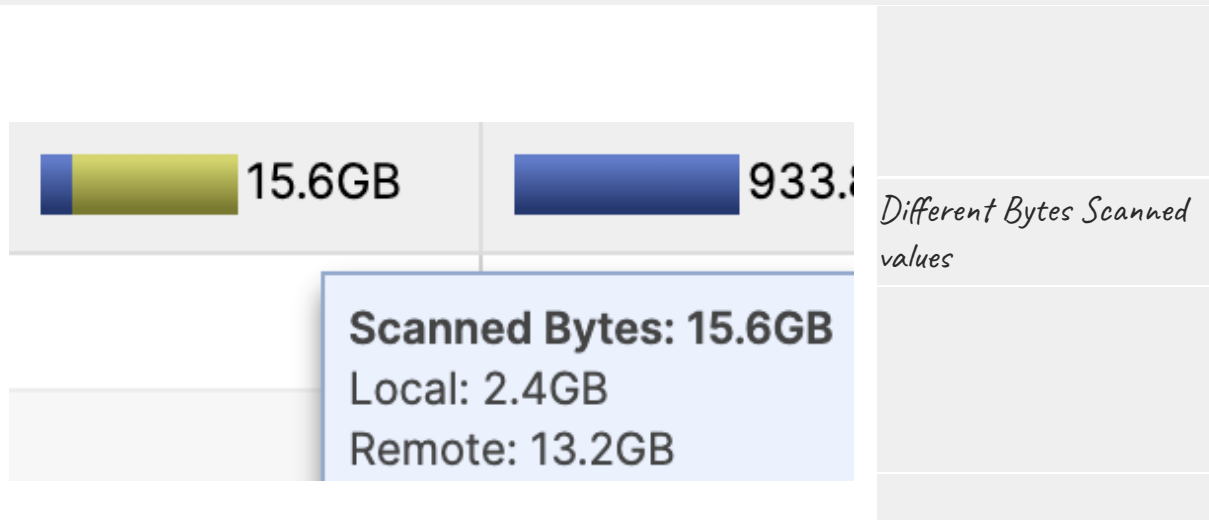
Warehouse cache example

Total duration → As we can see, the first query spent 7.53 minutes fetching all the data, as it's a big table with 933 million rows. The second case spends way less time as almost everything was re-used from the Warehouse cache. Different times are involved in getting the Total duration, like the Compilation Time, Queuing Time, and Execution Time.



Different times to calculate the Total Duration.

Bytes Scanned → We can also see that the number of Bytes Scanned was 15.6GB, and there are two colors. The blue one means the data is re-used from the warehouse, whereas the yellow one means that it's fetched from the Storage layer. Almost everything was taken from the Storage layer in the first query, whereas in the second case, almost everything was re-used from the warehouse. The bytes scanned size is also lower because fewer columns were fetched.



COMPLETE EXAMPLE WITH ALL THE SNOWFLAKE CACHE TYPES

Let's take one last example to see how it works. Let's go query by query:

First query: There is no information in the cache, so everything comes from the Long Term Storage (Storage Layer). We can see that with the Bytes Scanned row, as it's completely yellow.


```
SELECT * FROM ANALYTICS.PUBLIC.MYTABLE;
```

Cluster Number	Size ▼	Total Duration	Bytes Scanned	Rows
1	X-Small	897ms	3.3MB	4

Example not using Snowflake cache

Second query: We perform the same query. As less than 24 hours have been spent, this information is in the Query Result Cache, so we can see that we didn't scan bytes, and the duration is eight times lower than before.




```
SELECT * FROM ANALYTICS.PUBLIC.MYTABLE;
```

Cluster Number	Size ▼	Total Duration	Bytes Scanned	Rows
		 141ms		

Example of Query Result Cache in Snowflake

Third query: We access the same table, but do not run the same query. We need the column "MADEBY". We are not going to be able to fetch the information from the Query Result Cache, so we will fetch it from the Warehouse Cache. For that reason, the Bytes Scanned is not empty like before.

```
SELECT MADEBY FROM ANALYTICS.PUBLIC.MYTABLE;
```

Cluster Number	Size ▼	Total Duration	Bytes Scanned	Rows
1	X-Small	 179ms	 1.9MB	 4

Warehouse Cache example in Snowflake

Fourth query: We want to know the number of rows in "MYTABLE". This information is stored in the Metadata Cache, so we won't use any warehouse, having a tiny duration and no Bytes Scanned.

```
SELECT COUNT(*) FROM ANALYTICS.PUBLIC.MYTABLE;
```

Cluster Number	Size	Total Duration	Bytes Scanned	Rows
		193ms		

Metadata Cache Example

HOW TO IMPROVE SNOWFLAKE PERFORMANCE

After understanding all the types of cache we have in Snowflake, and how the warehouses work as we have seen in other chapters, I will give you some tips that will help us consume fewer credits and increase the performance of the queries.

1. **Use dedicated Virtual Warehouses** → It's good to have a Virtual Warehouse for each type of task. For example, a Virtual Warehouse for BI tasks, another for Data Science... As the users will perform similar queries, the results can be re-used easily.
2. **Scale UP/DOWN for workloads that are known** → If we know that Mondays at 10 a.m. the number of queries increases by x2, or we need to do a report that requires a lot of Snowflake power, we should scale up.
3. **Multi-Warehouses for unknown workloads** → Sometimes, the number of users increases without knowing that. For that reason, we can set up multi-warehouses.
4. Try to maximize the use of the cache.
5. **Cluster keys** → Use them in big tables to improve their performance, especially in columns that you use to filter (WHERE, JOINS...).

If you follow these tips, you'll see the results soon.

TYPICAL EXAM QUESTIONS

1. What are the different caching mechanisms available in Snowflake?

- 1. Metadata cache*
- 2. Query result cache*
- 3. Index cache*
- 4. Table cache*
- 5. Warehouse cache*

Solution: 1, 2, 5

2. A query executed a couple of hours ago, which spent more than 5 minutes to run, is executed again, and it returned the results in less than a second. What might have happened?

- 1. Snowflake used the persisted query results from the metadata cache*
- 2. Snowflake used the persisted query results from the query result cache*
- 3. Snowflake used the persisted query results from the warehouse cache*
- 4. A new Snowflake version has been released in the last two hours, improving the speed of the service*

Solution: 2

3. Are Snowflake caches automatically invalidated if the underlying data changes?

1. True
2. False

Solution: 1. If the data in the Storage Layer changes, the caches are automatically invalidated.

4. What command will you execute if you want to disable the query cache?

1. ALTER SESSION SET USE_CACHED_RESULT = TRUE
2. ALTER SESSION SET USE_CACHED_RESULT = FALSE
3. ALTER SESSION SET USE_CACHED_RESULT = ON
4. ALTER SESSION SET USE_CACHED_RESULT = OFF

Solution: 2

5. Which type of data incur in Snowflake storage cost?

1. Data Stored in permanent tables.
2. Data Stored in temporal tables.
3. Cache results.
4. Data retained for Fail-Safe & Time-Travel.

Solution: 1, 2, 4. This question is essential, and it already appeared in the Snowflake pricing chapter, but we must know that cache results do NOT incur Storage costs.

6. Which cache runs for 24 hours?

1. Metadata cache
2. Results cache
3. Warehouse cache

Solution: 2. Query Result cache is also known as Results Cache.

7. May the warehouse cache be reset if a running warehouse is suspended and resumes?

1. True
2. False

Solution: 1

8. Does the warehouse cache size change with the warehouse-size?

1. True
2. False

Solution: 1. The larger the warehouse is (the more servers it has), the larger the warehouse cache size is.

9. To improve the performance, which of the below techniques can be used in Snowflake?

1. *Cluster Keys*
2. *Multi-Warehouses*
3. *Maximize the cache use*
4. *Increasing the Warehouse Size*
5. *Dedicated Warehouses*

Solution: 1, 2, 3, 4, 5

Thanks for Reading!