Hive

1. Why hive metadata is stored in sql?

Hive Meta Store is a component of Apache Hive that provides a relational database to store Hive's metadata. It's responsible for keeping track of the Hive's data structures, such as tables, partitions and columns. It also enables Hive to use external databases, such as MySQL or PostgreSQL, to store its metadata, instead of using the default embedded Derby database.

1. Which sql is the default database for hive?

Hive doesn't have a default SQL database in the traditional sense, and its operations and querying are based on HiveQL, which is specific to Hive's ecosystem and its interaction with Hadoop-based storage

1. What is managed table?

Managed table is a type of table that is managed by the Hive metastore.

Managed tables in Hive have the following characteristics:

* **Data Management**: Hive takes care of organizing and storing the data for managed tables. The data is often stored in HDFS or other Hadoop-compatible storage systems.
* **Metadata Management**: Hive stores metadata about the managed table, such as its schema and other attributes, in the Hive metastore. The metastore is a database that keeps track of the structure and properties of Hive tables.
* **Data Consistency**: Since Hive manages both the metadata and the data, it ensures that the data in the table remains consistent with its schema. Inserting, updating, or deleting records is done through HiveQL queries, and Hive handles the data operations behind the scenes.
* **Table and Data Lifecycle**: Hive can control the lifecycle of the table and its data. For example, if you drop a managed table, Hive will remove both the metadata and the associated data.

1. What is external table?

External table is a type of table that is defined in Hive's metadata but does not manage the data itself. Instead, the data for an external table is stored externally in a location that you specify, and Hive simply provides a schema and metadata definition for querying that data using HiveQL.

1. When do we use external table?

* When you need to share data between multiple applications or systems. By using an external table, you can allow different tools or processes to access and work with the same dataset without duplicating the data or requiring them to go through Hive.
* When working with data that's generated by other systems, databases, or processes, using an external table can be a way to integrate that data into your Hive-based analytics workflow without the need to migrate it

1. Diff between managed and external table?

|  |  |
| --- | --- |
| **Managed Table** | **Extenal Table** |
| By default, Hive creates an Internal or Managed Table. | Use EXTERNAL option/clause to create an external table |
| Hive owns the metadata, table data by managing the lifecycle of the table | Hive manages the table metadata but not the underlying file. |
| Dropping an Internal table drops metadata from Hive Metastore and files from HDFS | Dropping an external table drops just metadata from Metastore without touching the actual file on HDFS/S3 |
| Metadata on Inserts, creation of new partitions, etc. are updated automatically during inserts through the metastore | You need to explicitly run sync\_partitions to sync changes on S3 with the metastore |
| Managed tables are suitable when you want Hive to take care of data management, need simplicity, and are less concerned with external data sources or sharing. | External tables are suitable when you need to work with data from external sources, share data between applications, maintain data control, or perform complex data integrations |

1. What happens if you don’t provide location to external table?

Error during the table creation, The location is a critical piece of information for external tables because it specifies where the actual data is stored. Without this information, Hive wouldn't know where to look for the data when you query the table

1. Performance optimization in hive?

**Data Format Selection**: Choosing optimized file formats like Parquet, ORC (Optimized Row Columnar), or Avro can significantly improve query performance. These formats are columnar and provide compression, predicate pushdown, and schema evolution benefits.

**Partitioning**: Partitioning divides data into smaller, manageable chunks based on specific columns. This can greatly improve query performance, as it reduces the amount of data scanned when filters are applied to queries

**Bucketing (Clustering)**: Bucketing or clustering involves dividing data into buckets based on hash values of specific columns. This can help evenly distribute data across buckets and improve join operations

**Indexing**: Hive supports indexing on certain columns using techniques like Bitmap Indexes or Compact Indexes. This can speed up lookups and filtering operations

**Caching**: Hive supports caching query results in memory. This is particularly useful for frequently executed queries, as it avoids recomputation

**Join Optimization**: Use broadcast joins for small tables and MapReduce-based joins for larger ones. Broadcast joins replicate the smaller table to all nodes, reducing data transfer overhead.

**Query Optimization**: Write efficient queries by avoiding unnecessary joins, aggregations, and Cartesian products. Optimize subqueries and use appropriate join strategies (e.g., broadcast, map-side, reduce-side) based on the dataset size

**Tuning Parameters**: Tune Hive configuration parameters like memory settings, parallelism, and query execution settings to match your cluster's resources and workload characteristics

1. Explain partition table. Give example

Partitioned table in Hive is a way of organizing data within a table by dividing it into subdirectories or partitions based on the values of one or more columns.

Eg: In the transaction table of sales creates,

CREATE TABLE sales\_transaction\_partitioned (

trans\_id INT,

amount INT )

PARTITIONED BY(year INT,month INT);

Based on the partitions run the query. Eg: select \* from sales\_transaction\_partitioned where year =2023 and month =10;

1. Explain bucket table. Give example

Bucketed table in Hive is a way of organizing data within a table by dividing it into a fixed number of buckets based on the hash values of one or more columns. Bucketing is used to evenly distribute data across a predefined number of files, which can help improve query performance, especially when performing join operations.

Eg: CREATE TABLE sales\_order\_bucket(

order\_id INT,

customer\_id INT,

amount INT )

CLUSTERED BY (customer\_id) INTO 4 BUCKETS

Based on the 4 buckets ,we can use join and other operations

1. Diff between partition and bucketed table.

Partition table: Based on values of columns of a table, Partition divides large amount of data into multiple slices. What that means is we are able to differentiate a large amount of data on the basis of our need, for example if we have the data for all the employees working in a particular company ( with huge number of employees) but we need to survey only the employees which belong to a particular category, in the absence of partitioning our process would be to scan through all the entries and find those out, but if we partition our table on the basis of category then it becomes very simple to survey the lot.

Bucketed table: Bucketing basically puts data into more manageable or equal parts. When we go for partitioning, we might end up with multiple small partitions based on column values. But when we go for bucketing, we restrict number of buckets to store the data. It is useful for the situation in which the field has high cardinality and data is evenly spread among all buckets