Hive Data Definition

Hive Data Definition Language (DDL) is a subset of Hive SQL statements that describe the data structure in Hive by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets. Most Hive DDL statements start with the keywords CREATE , DROP , or ALTER . The syntax of Hive DDL is very similar to the DDL in SQL. The comments in Hive start from –

SELECT statement

The most common use case of using Hive is to query the data in Hadoop. To achieve this,we need to write and execute the SELECT statement in Hive. The typical work done by the SELECT statement is to project the rows meeting query conditions specified in the WHERE clause after the target table and return the result set. The SELECT statement is quite often used with FROM , DISTINCT , WHERE , and LIMIT keywords

The JOIN statement

Hive JOIN is used to combine rows from two or more tables together. Hive supports common JOIN operations such as what's in the RDBMS, for example, JOIN , LEFT OUTER JOIN , RIGHT OUTER JOIN , FULL OUTER JOIN , and CROSS JOIN . However, Hive only supports equal JOIN instead of unequal JOIN , because unequal JOIN is difficult to be converted to MapReduce jobs.

The INNER JOIN in Hive uses JOIN keywords, which return rows meeting the JOIN conditions from both left and right tables. The INNER JOIN keyword can also be omitted by comma-separated table names since Hive 0.13.0.

Hive database

The database in Hive describes a collection of tables that are used for a similar purpose or belong to the same groups. If the database is not specified, the default database is used.

Whenever a new database is created, Hive creates a directory for each database at/user/hive/warehouse , defined in hive.metastore.warehouse.dir . For example, themyhivebook database is located at /user/hive/datawarehouse/myhivebook.db .

However, the default database doesn't have its own directory

Hive internal and external tables

The concept of a table in Hive is very similar to the table in the relational database. Each table associates with a directory configured in ${HIVE_HOME}/conf/hive-site.xml in HDFS. By default, it is /user/hive/warehouse in HDFS. For example,/user/hive/warehouse/employee is created by Hive in HDFS for the employee table. All the data in the table will be kept in the directory. The Hive table is also referred to as internal or managed tables.

```
CREATE TABLE emp_details
(
emp_name STRING,
unit STRING,
exp INT,
location STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

When there is data already in HDFS, an external Hive table can be created to describe the data. It is called EXTERNAL because the data in the external table is specified in the LOCATION properties instead of the default warehouse directory.

```
CREATE EXTERNAL TABLE weatherext ( wban INT, date STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘,’
LOCATION ‘ /hive/data/weatherext’;
```

When keeping data in the internal tables, Hive fully manages the life cycle of the table and data. This means the data is removed once the internal table is dropped. If the external table is dropped, the table metadata is deleted but the data is kept. Most of the time, an external table is preferred to avoid deleting data along with tables by mistake.

Hive partitions

By default, a simple query in Hive scans the whole Hive table. This slows down the performance when querying a large-size table. The issue could be resolved by creatingHive partitions, which is very similar to what’s in the RDBMS.

```
CREATE TABLE emp_details_partitioned
(
emp_name STRING,
unit STRING,
exp INT
)
PARTITIONED BY (location STRING);
```

In Hive, each partitioncorresponds to a predefined partition column(s) and stores it as a subdirectory in thetable's directory in HDFS. When the table gets queried, only the required partitions(directory) of data in the table are queried, so the I/O and time of query is greatly reduced.

Hive buckets

Besides partition, bucket is another technique to cluster datasets into more manageable parts to optimize query performance. Different from partition, the bucket corresponds to segments of files in HDFS.

For example, the employee_partitioned table from the previous section uses the year and month as the top-level partition. If there is a further request to use the employee_id as the third level of partition, it leads to many deep and small partitions and directories.

For instance, we can bucket the employee_partitioned table using employee_id as the bucket column. The value of this column will be hashed by a user-defined number into buckets. The records with the same employee_id will always be stored in the same bucket (segment of files). By using buckets, Hive can easily and efficiently do and map side.

Hive views

In Hive, views are logical data structures that can be used to simplify queries by either hiding the complexities such as joins, subqueries, and filters or by flatting the data. Unlike some RDBMS, Hive views do not store data or get materialized.

```
CREATE VIEW temperature_data_vw AS SELECT SUBSTR(full_date,7,4) AS
Year,MAX(temperature) from temperature_data GROUP BY SUBSTR(full_date,7,4) HAVING
COUNT(SUBSTR(full_date,7,4))>2;
```

Once the Hive view is created, its schema is frozen immediately. Subsequent changes to the underlying tables (for example, adding a column) will not be reflected in the view's schema. If an underlying table is dropped or changed, subsequent attempts to query the invalid view will fail

Hive Data Manipulations

The ability to manipulate data is a critical capability in big data analysis. Manipulating data is the process of exchanging, moving, sorting, and transforming the data. This technique is used in many situations, such as cleaning data, searching patterns, creating trends, and so on. Hive offers various query statements, keywords, operators, and functions to carry out data manipulation.

In this chapter, we will cover the following topics:

Data exchange using LOAD , INSERT , IMPORT , and EXPORT

Order and sort

Operators and functions

Transaction

Data exchange – LOAD

To move data in Hive, it uses the LOAD keyword. Move here means the original data is moved to the target table/partition and does not exist in the original place anymore. The following is an example of how to move data to the Hive table or partition from local or HDFS files.

The LOCAL keyword specifies where the files are located in the host. If the LOCAL keyword is not specified, the files are loaded from the full Uniform Resource Identifier (URI) specified after INPATH or the value from the fs.default.name Hive property by default. The path after INPATH can be a relative path or an absolute path. The path either points to a file or a folder (all files in the folder) to be loaded, but the subfolder is not allowed in the path specified.

If the data is loaded into a partition table, the partition column must be specified. The OVERWRITE keyword is used to decide whether to append or replace the existing data in the target table/partition.

Data exchange – INSERT

To extract the data from Hive tables/ partitions, we can use the INSERT keyword. Like RDBMS, Hive supports inserting data by selecting data from other tables. This is a very common way to populate a table from existing data. The basic INSERT statement has the same syntax as a relational database's INSERT . However, Hive has improved its INSERT statement by supporting OVERWRITE , multiple INSERT , dynamic partition INSERT , as well as using INSERT to files.

Data exchange – EXPORT and IMPORT

When working with Hive, sometimes we need to migrate data among different environments. Or we may need to back up some data. Since Hive 0.8.0, EXPORT andIMPORT statements are available to support the import and export of data in HDFS for data migration or backup/restore purposes.

The EXPORT statement will export both data and metadata from a table or partition.Metadata is exported in a file called _metadata . Data is exported in a subdirectory called data:

ORDER and SORT

Another aspect to manipulate data in Hive is to properly order or sort the data or result setsto clearly identify the important facts, such as top N values, maximum, minimum, and so on.

There are the following keywords used in Hive to order and sort data:

ORDER BY (ASC|DESC) : This is similar to the RDBMS ORDER BY statement. A sorted order is maintained across all of the output from every reducer. It performs the global sort using only one reducer, so it takes a longer time to return the result. Usage with LIMIT is strongly recommended for ORDER BY . When hive.mapred.mode = strict (by default, hive.mapred.mode = nonstrict) is set and we do not specify LIMIT , there are exceptions.

SORT BY (ASC|DESC) : This indicates which columns to sort when ordering the reducer input records. This means it completes sorting before sending data to thereducer. The SORT BY statement does not perform a global sort and only makes sure data is locally sorted in each reducer unless we set mapred.reduce.tasks=1 . In this case, it is equal to the result of ORDER BY .

Operators and functions

To further manipulate data, we can also use expressions, operators, and functions in Hiveto transform data. The Hive wiki :

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

has offered specifications for each expression and function, so we do not want to repeat all of them here except a few important usages or tips in this chapter.

Hive has defined relational operators, arithmetic operators, logical operators, complex type constructors, and complex type operators. For relational, arithmetic, and logical operators, they are similar to standard operators in SQL/Java. We do not repeat them again in this chapter.

The functions in Hive are categorized as follows:

Mathematical functions: These functions are mainly used to perform mathematicalcalculations, such as RAND() and E() .

Collection functions: These functions are used to find the size, keys, and values for complex types, such as SIZE(Array<T>) .

Type conversion functions: These are mainly CAST and BINARY functions to convert one type to the other.

Date functions: These functions are used to perform date-related calculations, suchas YEAR(string date) and MONTH(string date) .

Conditional functions: These functions are used to check specific conditions with adefined value returned, such as COALESCE , IF , and CASE WHEN .

String functions: These functions are used to perform string-related operations, such as UPPER(string A) and TRIM(string A) .

Aggregate functions: These functions are used to perform aggregation (which isintroduced in the next chapter for more details), such as SUM() , COUNT(*) .

Table-generating functions: These functions transform a single input row intomultiple output rows, such as EXPLODE(MAP) and JSON_TUPLE(jsonString, k1, k2,…) .

Customized functions: These functions are created by Java code as extensions forHive.

Transactions

Before Hive version 0.13.0, Hive does not support row-level transactions. As a result, there is no way to update, insert, or delete rows of data. Hence, data overwrite can only happen on tables or partitions. This makes Hive very difficult when dealing with concurrent read/write and data-cleaning use cases.

Since Hive version 0.13.0, Hive fully supports row-level transactions by offering fullAtomicity, Consistency, Isolation, and Durability (ACID) to Hive. For now, all thetransactions are autocommuted and only support data in the Optimized Row Columnar (ORC) file (available since Hive 0.11.0) format and in bucketed tables.

HiveQL(Hive Query Language)?

Hive provides a CLI to write Hive queries using Hive Query Language (HiveQL). Generally HQL syntax is similar to the SQL syntax that most data analysts are familiar with.

Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, to ease learning.

Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language (Hive-QL).

Generally, HiveQL syntax is similar to the SQL syntax that most data analysts are familiar with. Hive supports four file formats those are TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).

- For single user metadata storage Hive uses derby database and
- For multiple user Metadata or shared Metadata case Hive uses MYSQL

HiveQL Manipulations

HiveQL data manipulation language parts is used to put data into tables and to extract data from tables to the filesystem.

It mainly uses SELECT ... WHERE clauses extensively when we discuss populating tables with data queried from other tables

Loading Data into Managed Tables

Since Hive has no row-level insert, update, and delete operations, the only way to put data into an table is to use one of the “bulk” load operations. Or you can just write files in the correct directories by other means.The below command is used to Load data into managed table

```
LOAD DATA
LOCAL INPATH '/home/acadgild/hive/complex_data_type.txt'
INTO TABLE complex_data_type;
```

Inserting Data into Tables from Queries

The INSERT statement lets you load data into a table from a query.

```
INSERT OVERWRITE TABLE emp_details_partitioned
```

```
partition(location='BBSR')
SELECT emp_name,unit,exp from emp_details where location='BBSR';
```

Dynamic Partition Inserts

If you have a lot of partitions to create, you have to write a lot of SQL! Fortunately, Hive also supports a dynamic partition feature, where it can infer the partitions to create based on query parameters.

```
CREATE TABLE emp_details_unit
(
emp_name STRING,
exp INT,
location STRING
)
PARTITIONED BY (unit STRING);
```

Built-in operators

Hive provides Built-in operators for Data operations to be implemented on the tables present inside Hive warehouse.

These operators are used for mathematical operations on operands, and it will return specific value as per the logic applied.

Types of Built-in Operators in HIVE are:

- Relational Operators
- Arithmetic Operators
- Logical Operators
- Operators on Complex types
- Complex type Constructors

Relational Operators:

We use Relational operators for relationship comparisons between two operands.

- Operators such as equals, Not equals, less than, greater than …etc
- The operand types are all number types in these Operators.

Arithmetic Operators:

We use Arithmetic operators for performing arithmetic operations on operands

- Arithmetic operations such as addition, subtraction, multiplication and division between operands we use these Operators.
- The operand types all are number types in these Operators

Logical Operators:

We use Logical operators for performing Logical operations on operands

- Logical operations such as AND, OR, NOT between operands we use these Operators.
- The operand types all are BOOLEAN type in these Operators

The following Table will give us details about Logical operators

Operators on Complex types:

The following Table will give us details about Complex Type Operators . These are operators which will provide a different mechanism to access elements in complex types.

Operators	**Operands**	**Description**
A[n]	A is an Array and n is an integer type	It will return nth element in the array A. The first element has index of 0
M[key]	M is a Map<K, V> and key has type K	It will return the values belongs to the key in the map

Complex type Constructors:

The following Table will give us details about Complex type Constructors. It will construct instances on complex data types. These are of complex data types such as Array, Map and Struct types in Hive.

In this section, we are going to see the operations performed on Complex type Constructors.

Operators	Operands	Description
array	(val1, val2, ...)	It will create an array with the given elements as mentioned like val1, val2
Create_ union	(tag, val1, val2, ...)	It will create a union type with the values that is being mentioned to by the tag parameter
map	(key1, value1, key2, value2, ...)	It will create a map with the given key/value pairs mentioned in operands
Named_struct	(name1, val1, name2, val2, ...)	It will create a Struct with the given field names and values mentioned in operands
STRUCT	(val1, val2, val3, ...)	Creates a Struct with the given field values. Struct field names will be col1, col2, .