北风网 IBEIFENG.COM

大数据Hadoop高薪直通车课程

Hive 深入使用

讲师：轩宇（北风网版权所有)

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

DataBase

Create/Drop/Alter Database

Create Database

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
  [COMMENT database_comment]
  [LOCATION hdfs_path]
  [WITH DBPROPERTIES (property_name=property_value, ...)];
```

The uses of SCHEMA and DATABASE are interchangeable – they mean the same thing. CREATE DATABASE was added in Hive 0.6 (HIVE-675). The WITH DBPROPERTIES clause was added in Hive 0.7 (HIVE-1836).

Drop Database

```
DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
```

The uses of SCHEMA and DATABASE are interchangeable – they mean the same thing. DROP DATABASE was added in Hive 0.6 (HIVE-675).

Create/Drop/Alter Database

Alter Database

```
ALTER (DATABASE|SCHEMA) database_name SET DBPROPERTIES (property_name=propert

ALTER (DATABASE|SCHEMA) database_name SET OWNER [USER|ROLE] user_or_role;
```

The uses of SCHEMA and DATABASE are interchangeable – they mean the same thing. ALTER SCHEMA was added in Hive 0.14 (HIVE-6601).

No other metadata about a database can be changed.

Use Database

```
USE database_name;
USE DEFAULT;
```

USE sets the current database for all subsequent HiveQL statements. To revert to the default database, use the keyword "`default`" instead of a database name.

`USE database_name` was added in Hive 0.6 (HIVE-675).

Show Database

Show Databases

```
SHOW (DATABASES|SCHEMAS) [LIKE 'identifier_with_wildcards'];
```

SHOW DATABASES or SHOW SCHEMAS lists all of the databases defined in the metastore. The uses of SCHEMAS and DATABASES are interchangeable – they mean the same thing.

The optional LIKE clause allows the list of databases to be filtered using a regular expression. Wildcards in the regular expression can only be '*' for any character(s) or '|' for a choice. Examples are 'employees', 'emp*', 'emp*|*ees', all of which will match the database named 'employees'.

Describe Database

Describe Database

> **Version information**
> As of Hive 0.7.

```
DESCRIBE DATABASE [EXTENDED] db_name;
DESCRIBE SCHEMA [EXTENDED] db_name;     -- (Note: Hive 0.15.0 and later)
```

DESCRIBE DATABASE shows the name of the database, its comment (if one has been set), and its root location on the filesystem. The uses of SCHEMA and DATABASE are interchangeable – they mean the same thing. DESCRIBE SCHEMA is added in Hive 0.15 (HIVE-8803).

EXTENDED also shows the database properties.

Table

Create/Drop/Truncate Table

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-Create/Drop/TruncateTable

```
## 创建表,指定表的类型,表的名称
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  ### 表的列,包含列名称,列类型
  [(col_name data_type [COMMENT col_comment], ...)]
  ### 表的说明
  [COMMENT table_comment]
  ### 分区表,指定分区的列名称,列类型
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  ### 数据文件中数据存储格式,每行分隔,每列分隔,数据文件类型
  [
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  ### 数据文件存储在HDFS上的位置
  [LOCATION hdfs_path]
  ### 表的属性设置
  [TBLPROPERTIES (property_name=property_value, ...)]
  ### 子查询
  [AS select_statement];
```

Create/Drop/Truncate Table

```
## 创建表,指定表的类型,表的名称
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  ### 指定表结构相同的表名称
  LIKE existing_table_or_view_name
  ### 数据文件存储在HDFS上的位置
  [LOCATION hdfs_path];

## 表中列的数据类型
data_type
  : primitive_type
  | array_type
  | map_type
  | struct_type
  | union_type
```

Create/Drop/Truncate Table

```
### 数据文件中数据存储格式
row_format
   : DELIMITED
     #### 每列数据分隔符
     [FIELDS TERMINATED BY char ]
     #### 每行数据分隔符
     [LINES TERMINATED BY char]

### 数据文件存储格式
file_format:
   : SEQUENCEFILE
   | TEXTFILE
   | RCFILE
   | ORC
   | PARQUET
   | INPUTFORMAT input_format_classname
     OUTPUTFORMAT output_format_classname
```

Create Table

员工表：

```
create table emp(
empno int,
ename string,
job string,
mgr int,
hiredate string,
sal double,
comm double,
deptno int
)
row format delimited fields terminated by '\t';
```

部门表：

```
create table dept(
deptno int,
dname string,
loc string
)
row format delimited fields terminated by '\t';
```

Load Data

LOAD DATA [LOCAL] **INPATH** 'filepath' [OVERWRITE] **INTO TABLE** tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

- LOCAL：从本地文件加载数据到hive表；否则从HDFS加载数据到hive表；
- OVERWRITE：是否覆盖表中已有数据；

```
load data local inpath '/home/hadoop/data/emp.txt' overwrite into table emp;
load data local inpath '/home/hadoop/data/dept.txt' overwrite into table dept;
```

Create Table As Select (CTAS)

Create Table As Select (CTAS)

Tables can also be created and populated by the results of a query in one create-table-as-select (CTAS) statement. The table created by CTAS is atomic, meaning that the table is not seen by other users until all the query results are populated. So other users will either see the table with the complete results of the query or will not see the table at all.

There are two parts in CTAS, the SELECT part can be any SELECT statement supported by HiveQL. The CREATE part of the CTAS takes the resulting schema from the SELECT part and creates the target table with other table properties such as the SerDe and storage format.

Example:

```
CREATE TABLE new_key_value_store
   ROW FORMAT SERDE "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe"
   STORED AS RCFile
   AS
SELECT (key % 1024) new_key, concat(key, value) key_value_pair
FROM key_value_store
SORT BY new_key, key_value_pair;
```

Create Table Like

Create Table Like

The LIKE form of CREATE TABLE allows you to copy an existing table definition exactly (without copying its data). In contrast to CTAS, the statement below creates a new empty_key_value_store table whose definition exactly matches the existing key_value_store in all particulars other than table name. The new table contains no rows.

```
CREATE TABLE empty_key_value_store
LIKE key_value_store;
```

Before Hive 0.8.0, CREATE TABLE LIKE view_name would make a copy of the view. In Hive 0.8.0 and later releases, CREATE TABLE LIKE view_name creates a table by adopting the schema of view_name (fields and partition columns) using defaults for SerDe and file formats.

Create/Drop/Truncate Table

Truncate Table

Version information
As of Hive 0.11.0 (HIVE-446).

```
TRUNCATE TABLE table_name [PARTITION partition_spec];

partition_spec:
  : (partition_column = partition_col_value, partition_column = partition_col
```

Removes all rows from a table or partition(s). Currently target table should be native/managed table or exception will be thrown. User can specify partial partition_spec for truncating multiple partitions at once and omitting partition_spec will truncate all partitions in the table.

Create/Drop/Truncate Table

Drop Table

```
DROP TABLE [IF EXISTS] table_name [PURGE];    -- (Note: PURGE available in H
```

DROP TABLE removes metadata and data for this table. The data is actually moved to the .Trash/Current directory if Trash is configured (and PURGE is not specified). The metadata is completely lost.

When dropping an EXTERNAL table, data in the table will *NOT* be deleted from the file system.

External Tables

- 内部表也称之为**MANAGED_TABLE**；
- 默认存储在/user/hive/warehouse下，也可以通过location指定；
- **删除表时，会删除表数据以及元数据**；

- 外部表称之为**EXTERNAL_TABLE**；
- 在创建表时可以自己指定目录位置(LOCATION)；
- **删除表时，只会删除元数据不会删除表数据**；

External Tables

The EXTERNAL keyword lets you create a table and provide a LOCATION so that Hive does not use a default location for this table. This comes in handy if you already have data generated. When dropping an EXTERNAL table, data in the table is *NOT* deleted from the file system.

An EXTERNAL table points to any HDFS location for its storage, rather than being stored in a folder specified by the configuration property `hive.metastore.warehouse.dir`.

```
CREATE EXTERNAL TABLE page_view(viewTime INT, userid BIGINT,
     page_url STRING, referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User',
     country STRING COMMENT 'country of origination')
 COMMENT 'This is the staging page view table'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY '\054'
 STORED AS TEXTFILE
 LOCATION '<hdfs_location>';
```

You can use the above statement to create a page_view table which points to any hdfs location for its storage. But you still have to make sure that the data is delimited as specified in the CREATE statement above.

Partitioned Tables

分区表实际上就是对应一个HDFS文件系统上的独立的文件夹，该文件夹下是该分区所有的数据文件。Hive中的分区就是分目录，把一个大的数据集根据业务需要分割成更下的数据集。

在查询时通过WHERE子句中的表达式来选择查询所需要的指定的分区，这样的查询效率会提高很多。

```
create table dept_partition(
deptno int,dname string,loc string
)
PARTITIONED BY (event_month string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

Partitioned Tables

Partitioned tables can be created using the PARTITIONED BY clause. A table can have one or more partition columns and a separate data directory is created for each distinct value combination in the partition columns. Further, tables or partitions can be bucketed using CLUSTERED BY columns, and data can be sorted within that bucket via SORT BY columns. This can improve performance on certain kinds of queries.

If, when creating a partitioned table, you get this error: "FAILED: Error in semantic analysis: Column repeated in partitioning columns," it means you are trying to include the partitioned column in the data of the table itself. You probably really do have the column defined. However, the partition you create makes a pseudocolumn on which you can query, so you must rename your table column to something else (that users should not query on!).

Partitioned Tables

Here is an example. Suppose your original table was this:

```
id      int,
date    date,
name    varchar
```

Now you want to partition on date. Your Hive definition would be this:

```
create table table_name (
  id                int,
  dtDontQuery       string,
  name              string
)
partitioned by (date string)
```

Now your users will still query on "where date = '...'" but the 2nd column will be the original values.

Partitioned Tables

Here's an example statement to create a table:

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
     page_url STRING, referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User')
 COMMENT 'This is the page view table'
 PARTITIONED BY(dt STRING, country STRING)
 STORED AS SEQUENCEFILE;
```

The statement above creates the page_view table with viewTime, userid, page_url, referrer_url, and ip columns (including comments). The table is also partitioned and data is stored in sequence files. The data format in the files is assumed to be field-delimited by ctrl-A and row-delimited by newline.

Partitioned Tables

The statement above creates the page_view table with viewTime, userid, page_url, referrer_url, and ip columns (including comments). The table is also partitioned and data is stored in sequence files. The data format in the files is assumed to be field-delimited by ctrl-A and row-delimited by newline.

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
     page_url STRING, referrer_url STRING,
     ip STRING COMMENT 'IP Address of the User')
 COMMENT 'This is the page view table'
 PARTITIONED BY(dt STRING, country STRING)
 ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\001'
STORED AS SEQUENCEFILE;
```

The above statement lets you create the same table as the previous table.

In the previous examples the data is stored in <hive.metastore.warehouse.dir>/page_view. Specify a value for the key hive.metastore.warehouse.dir in Hive config file hive-site.xml.

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

Hive Data Types

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

Numeric Types

- TINYINT (1-byte signed integer, from -128 to 127)
- SMALLINT (2-byte signed integer, from -32,768 to 32,767)
- INT (4-byte signed integer, from -2,147,483,648 to 2,147,483,647)
- BIGINT (8-byte signed integer, from -9,223,372,036,854,775,808 to 9
- FLOAT (4-byte single precision floating point number)
- DOUBLE (8-byte double precision floating point number)
- DECIMAL
    - Introduced in Hive 0.11.0 with a precision of 38 digits
    - Hive 0.13.0 introduced user definable precision and scale

Date/Time Types

- TIMESTAMP (Note: Only available starting with Hive 0.8.0)
- DATE (Note: Only available starting with Hive 0.12.0)

Hive Data Types

String Types

- STRING
- VARCHAR (Note: Only available starting with Hive 0.12.0)
- CHAR (Note: Only available starting with Hive 0.13.0)

Misc Types

- BOOLEAN
- BINARY (Note: Only available starting with Hive 0.8.0)

Complex Types

- arrays: ARRAY<data_type> (Note: negative values and non-constant expressions are allowed as of Hive 0.14.)
- maps: MAP<primitive_type, data_type> (Note: negative values and non-constant expressions are allowed as of Hive 0.14.)
- structs: STRUCT<col_name : data_type [COMMENT col_comment], ...>
- union: UNIONTYPE<data_type, data_type, ...> (Note: Only available starting with Hive 0.7.0.)

Primitive Types

- Types are associated with the columns in the tables. The following Primitive types are supported:
- Integers
  - TINYINT - 1 byte integer
  - SMALLINT - 2 byte integer
  - INT - 4 byte integer
  - BIGINT - 8 byte integer
- Boolean type
  - BOOLEAN - TRUE/FALSE
- Floating point numbers
  - FLOAT - single precision
  - DOUBLE - Double precision
- String type
  - STRING - sequence of characters in a specified character set

https://cwiki.apache.org/confluence/display/Hive/Tutorial

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

Hive Data Manipulation

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

There are multiple ways to modify data in Hive:

- LOAD
- INSERT
  - into Hive tables from queries
  - into directories from queries
  - into Hive tables from SQL
- UPDATE
- DELETE

EXPORT and IMPORT commands are also available (as of Hive 0.8).

Loading files into tables

Hive does not do any transformation while loading data into tables. Load operations are currently pure copy/move operations that move datafiles into locations corresponding to Hive tables.

```
LOAD DATA [LOCAL] INPATH 'filepath'
[OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

Loading files into tables

- *filepath* can be:
  - a relative path, such as `project/data1`
  - an absolute path, such as `/user/hive/project/data1`
  - a full URI with scheme and (optionally) an authority, such as `hdfs://namenode:9000/user/hive/project/data1`
- The target being loaded to can be a table or a partition. If the table is partitioned, then one must specify a specific partition of the table by specifying values for all of the partitioning columns.
- *filepath* can refer to a file (in which case Hive will move the file into the table) or it can be a directory (in which case Hive will move all the files within that directory into the table). In either case, *filepath* addresses a set of files.
- If the keyword LOCAL is specified, then:
  - the load command will look for *filepath* in the local file system. If a relative path is specified, it will be interpreted relative to the user's current working directory. The user can specify a full URI for local files as well - for example: `file:///user/hive/project/data1`
  - the load command will try to copy all the files addressed by *filepath* to the target filesystem. The target file system is inferred by looking at the location attribute of the table. The copied data files will then be moved to the table.

Loading files into tables

- If the keyword LOCAL is *not* specified, then Hive will either use the full URI of *filepath*, if one is specified, or will apply the following rules:
  - If scheme or authority are not specified, Hive will use the scheme and authority from the hadoop configuration variable `fs.default.name` that specifies the Namenode URI.
  - If the path is not absolute, then Hive will interpret it relative to `/user/<username>`
  - Hive will *move* the files addressed by *filepath* into the table (or partition)
- If the OVERWRITE keyword is used then the contents of the target table (or partition) will be deleted and replaced by the files referred to by *filepath*; otherwise the files referred by *filepath* will be added to the table.
  - Note that if the target table (or partition) already has a file whose name collides with any of the filenames contained in *filepath*, then the existing file will be replaced with the new file.

Loading files into tables

Notes

- *filepath* cannot contain subdirectories.
- If the keyword LOCAL is not given, *filepath* must refer to files within the same filesystem as the table's (or partition's) location.
- Hive does some minimal checks to make sure that the files being loaded match the target table. Currently it checks that if the table is stored in sequencefile format, the files being loaded are also sequencefiles, and vice versa.
- A bug that prevented loading a file when its name includes the "+" character is fixed in release 0.13.0 (HIVE-6048).
- Please read CompressedStorage if your datafile is compressed.

Hive Data Manipulation

- 加载本地文件到hive表
- 加载HDFS文件到hive表
- 覆盖表中已有数据
- 创建表时通过select加载
- 创建表时通过insert加载
- 创建表时通过location指定加载

Hive Data Manipulation

- 通过insert…directory导出
- 通过hadoop命令导出
- 通过hive shell命令 + 管道(hive -f/-e | sed/grep/awk > file)
- 使用**sqoop**

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

Queries

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

- Select Syntax
  - WHERE Clause
  - ALL and DISTINCT Clauses
  - Partition Based Queries
  - HAVING Clause
  - LIMIT Clause
  - REGEX Column Specification
  - More Select Syntax
    - GROUP BY
    - SORT BY, ORDER BY, CLUSTER BY, DISTRIBUTE BY
    - JOIN
    - UNION
    - TABLESAMPLE
    - Subqueries
    - Virtual Columns
    - Operators and UDFs
    - LATERAL VIEW
    - Windowing, OVER, and Analytics
    - Common Table Expressions

Queries

SELECT [ALL | DISTINCT] select_expr, select_expr, ...

FROM table_reference

[WHERE where_condition]

[GROUP BY col_list]

[CLUSTER BY col_list

| **[DISTRIBUTE BY col_list] [SORT BY col_list]**

]

[LIMIT number]

Queries

- 全表查询、指定字段查询
- = />= / <= /between and /limit
- (not) in / is (not) null
- max/min/count/sum/avg
- group by / having
- join

Queries

Sort

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

- Order By

  全局排序，一个Reduce

- Sort By

  每个reduce内部进行排序，全局不是排序

- Distribute By

  类似MR中partition，进行分区，结合sort by使用

- Cluster By

  当distribute和sort字段相同时，使用方式

课程大纲

1. Hive 表的创建
2. Hive 数据类型
3. Hive 数据迁移
4. Hive 常见查询
5. Hive UDF编程

Hive Operators and User-Defined Functions

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

In Beeline or the CLI, use the commands below to show the latest documentation:

```
SHOW FUNCTIONS;
DESCRIBE FUNCTION <function_name>;
DESCRIBE FUNCTION EXTENDED <function_name>;
```

User-Defined Functions

Hive自带了一些函数，比如:max /min等，但是数量有限，自己可以通过自定义UDF来方便的扩展。

UDF：用户自定义函数，允许用户扩展HiveQL功能；

- UDF(User-Defined-Function)
  一进一出

- UDAF(User-Defined Aggregation Funcation)
  聚集函数，多进一出；
  类似于：count/max/min

- UDTF(User-Defined Table-Generating Functions)
  一进多出；
  如lateral view explore()

User-Defined Functions

https://cwiki.apache.org/confluence/display/Hive/HivePlugins

编程步骤：
1、继承org.apache.hadoop.hive.ql.UDF
2、需要实现evaluate函数；evaluate函数支持**重载**；

注意事项：
1、UDF**必须要有返回类型**，可以返回null,但是返回类型不能为void；
2、UDF中**常用Text/LongWritable等类型**，不推荐使用java类型；

本课程版权归北风网所有

欢迎访问我们的官方网站

www.ibeifeng.com