

Hive SQL 语法大全

基于语法描述说明

```
CREATE DATABASE [IF NOT EXISTS] db_name [LOCATION] 'path';  
SELECT expr, ... FROM tbl ORDER BY col_name [ASC | DESC]  
(A | B | C)
```

如上语法，在语法描述中出现：

- `[]`，表示可选，如上 `[LOCATION]` 表示可写、可不写
- `|`，表示或，如上 `ASC | DESC`，表示二选一
- `...`，表示序列，即未完结，如上 `SELECT expr, ...` 表示在SELECT后可以跟多个 `expr`（查询表达式），以逗号隔开
- `()`，表示必填，如上 `(A | B | C)` 表示此处必填，填入内容在A、B、C中三选一

数据库操作

创建数据库

```
create database [if not exists] db_name [location 'path'] [COMMENT database_comment];
```

- `[location]`，自定义数据库存储位置，如不填写，默认数据库在HDFS的路径为： `/user/hive/warehouse`
- `[COMMENT database_comment]`，可选，数据库注释

删除数据库

```
DROP DATABASE [IF EXISTS] db_name [CASCADE];
```

- `[CASCADE]`，可选，级联删除，即数据库内存在表，使用CASCADE可以强制删除数据库

数据库修改LOCATION

```
ALTER DATABASE database_name SET LOCATION hdfs_path;
```

不会在HDFS对数据库所在目录进行改名，只是修改location后，新创建的表在新的路径，旧的不变

选择数据库

```
USE db_name;
```

- 选择数据库后，后续 SQL 操作基于当前选择的库执行
- 如不使用use，默认在 default 库执行

查看数据库详细信息

```
desc database myhive;      #默认数据库的存放路径是HDFS的：/user/hive/warehouse内
```

查询当前USE的数据库

```
SELECT current_database();
```

表操作

数据类型

分类	类型	描述	字面量示例
原始类型	BOOLEAN	true/false	TRUE
	TINYINT	1字节的有符号整数 -128~127	1Y
	SMALLINT	2个字节的有符号整数, -32768~32767	1S
	INT	4个字节的带符号整数	1
	BIGINT	8字节带符号整数	1L
	FLOAT	4字节单精度浮点数1.0	

分类	类型	描述	字面量示例
	DOUBLE	8字节双精度浮点数	1.0
	DEICIMAL	任意精度的带符号小数	1.0
	STRING	字符串, 变长	"a","b"
	VARCHAR	变长字符串	"a","b"
	CHAR	固定长度字符串	"a","b"
	BINARY	字节数组	
	TIMESTAMP	时间戳, 毫秒值精度	122327493795
	DATE	日期	'2016-03-29'
		时间频率间隔	
复杂类型	ARRAY	有序的的同类型的集合	array(1,2)
	MAP	key-value,key必须为原始类型, value可以任意类型	map('a',1,'b',2)
	STRUCT	字段集合,类型可以不同	struct('1',1,1.0), named_struct('col1','1','col2',1,'clo3',1.0)
	UNION	在有限取值范围内的一个值	create_union(1,'a',63)

基础建表

```
CREATE [EXTERNAL] TABLE tb_name
    (col_name col_type [COMMENT col_comment], ..... )
    [COMMENT tb_comment]
    [PARTITIONED BY(col_name, col_type, .....)]
    [CLUSTERED BY(col_name, col_type, ..... ) INTO num BUCKETS]
    [ROW FORMAT DELIMITED FIELDS TERMINATED BY '']
    [LOCATION 'path']
```

- [EXTERNAL], 外部表: 并非Hive拥有的表, 只是临时关联数据去使用。

可以先有表, 然后把数据移动到表指定的LOCATION中
也可以先有数据, 然后创建表通过LOCATION指向数据

元数据: 历史记录

数据: 核心数据

外部表: 外包, 删元数据

内部表: 正式员工, 删元数据和数据

- [ROW FORMAT DELIMITED FIELDS TERMINATED BY ''] 指定列分隔符

```
CREATE EXTERNAL TABLE test_ext(id int) COMMENT 'external table' ROW FORMAT
DELIMITED FIELDS TERMINATED BY '\t' LOCATION
'hdfs://node1:8020/tmp/test_ext';      表示以\t分隔
```

- [PARTITIONED BY(col_name, col_type,)] 基于列分区

```
-- 分区表示意
CREATE TABLE test_ext(id int) COMMENT 'partitioned table' PARTITION BY(year
string, month string, day string) ROW FORMAT DELIMITED FIELDS TERMINATED BY
'\t';
```

- [CLUSTERED BY(col_name, col_type,)] 基于列分桶

```
CREATE TABLE course (c_id string,c_name string,t_id string) CLUSTERED BY(c_id)
INTO 3 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

数据分隔符：

数据在HDFS上以明文文件存在的，但是cat表文件显示数据挤在一起，vim会发现有特殊字符
这些特殊字符是分隔符。

基于其它表的结构建表

```
CREATE TABLE tbl_name LIKE other_tbl;
```

基于查询结果建表

```
CREATE TABLE tbl_name AS SELECT ...;
```

删除表

```
DROP TABLE tbl;
```

修改表

重命名

```
ALTER TABLE old RENAME TO new;
```

查看表类型

```
desc formatted stu;
```

修改属性

```
alter table tb set tblproperties(key=value);
-- 常用属性
("EXTERNAL"="TRUE") -- 内部表转外部表
('comment' = new_comment) -- 修改表注释
-- 其余属性参见
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-listTableProperties
```

分区操作

创建分区表

```
-- 分区表示意
CREATE TABLE test_ext(id int) COMMENT 'partitioned table' PARTITION BY(year string,
month string, day string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

添加分区

```
ALTER TABLE tablename ADD PARTITION (partition_key='partition_value', .....);
```

修改分区值

```
ALTER TABLE tablename PARTITION (partition_key='old_partition_value') RENAME TO
PARTITION (partition_key='new_partition_value');
```

注意

只会在元数据中修改，不会同步修改HDFS路径吗，如：

- 原分区路径为： `/user/hive/warehouse/test.db/test_table/month=201910`，分区名：`month='201910'`
- 将分区名修改为： `201911` 后，分区所在路径不变，依旧是：`/user/hive/warehouse/test.db/test_table/month=201910`

如果希望修改分区名后，同步修改HDFS的路径，并保证正常可用，需要：

- 在元数据库中：找到SDS表 -> 找到LOCATION列 -> 找到对应分区的路径记录进行修改
 - 如将记录的：`/user/hive/warehouse/test.db/test_table/month=201910` 修改为：`/user/hive/warehouse/test.db/test_table/month=201911`
- 在HDFS中，同步修改文件夹名

- 如将文件夹: `/user/hive/warehouse/test.db/test_table/month=201910` 修改为: `/user/hive/warehouse/test.db/test_table/month=201911`

删除分区

```
ALTER TABLE tablename DROP PARTITION (partition_key='partition_value');
```

删除分区后, 只是在元数据中删除, 即删除元数据库中:

- `PARTITION` 表
- `SDS` 表

相关记录

分区所在的HDFS文件夹依旧保留

加载数据

LOAD DATA

```
LOAD DATA [LOCAL] INPATH 'path' INTO TABLE tbl  
partition(partition_key='partition_value');
```

INSERT SELECT

```
INSERT (OVERWRITE | INTO) TABLE tbl PARTITION(partition_key='partition_value')  
SELECT ... FROM ...;
```

分桶操作

建表

```
CREATE TABLE course (c_id string,c_name string,t_id string)  
[partition(key='value')]  
clustered by(c_id) into 3 buckets row FORMAT DELIMITED FIELDS TERMINATED by '\t';
```

- `CLUSTERED BY(col)` 指定分桶列
- `INTO 3 BUCKETS`, 设定3个桶

分桶表的自动优化:

```
set hive.enforce.bucketing=true;
```

设置自动匹配桶数量的reduces task数量

数据加载

```
INSERT (OVERWRITE | INTO) TABLE tbl
    [PARTITION(partition_key='partition_value')]
    SELECT ... FROM ... CLUSTER BY(col);
```

分区表：根据不同的分区项进行分表

分桶表：将一个表进行hash取模分配到固定数量的表中

分桶表无法使用LOAD DATA进行数据加载，他只是简单的移动数据，不会触发MapReduce

数据加载

LOAD DATA

将数据文件加载到表

```
load data [local] INPATH 'path' [overwrite|into] TABLE tbl
    [PARTITION(partition_key='partition_value')];    -- 指定分区可选
```

使用local，数据不在HDFS

不使用local，数据在hdfs

注意：基于hdfs进行load加载数据，源数据文件会消失（本质mv）

INSERT SELECT

将其它表数据，加载到目标表

```
INSERT (OVERWRITE | INTO) TABLE tbl
    [PARTITION(partition_key='partition_value')]    -- 指定分区，可选
    SELECT ... FROM ... [CLUSTER BY(col)];          -- 指定分桶列，可选
```

数据导出

INSERT OVERWRITE SELECT

```
INSERT OVERWRITE [LOCAL] DIRECTORY 'path'    -- LOCAL可选，带LOCAL导出Linux
本地，不带LOCAL导出到HDFS
    [ROW FORMAT DELIMITED FIELDS TERMINATED BY '']    -- 可选，自定义列分隔符
    SELECT ... FROM ...;
```

bin/hive

- bin/hive -e 'sql语句' > export_file -e：执行sql语句，结果重定向到导出文件中
- bin/hive -f 'sql脚本文件' > export_file -f：执行sql脚本文件，执行的结果重定向到导出文件中

复杂类型

类型	定义	示例	内含元素类型	元素个数	取元素	可用函数
array	array<类型>	如定义为 array数据为：1,2,3,4,5	单 值， 类型 取决于定义	动 态， 不限 制	array[数 字序号] 序号从0 开始	size(n)统计元素个数 array_contains(n,v值)判 断是否包含指定数据 下标索引
map	map<kv>	如定义为： map<string, int>数据为： {'a': 1, 'b': 2, 'c': 3}	键值 对， K- V	动 态， 不限 制	map[key] 取出对应 key的 value	size统计元素个数 array_contains判断是否 包含指定数据 map_keys 取出全部key，返回 array map_values取出 全部values，返回array
struct	struct<子 列名类 型, 子列 名类 型...>	如定义为： struct<c1 string, c2 int, c3 date>数据 为：'a', 1, '2000-01-01'	单 值， 类型 取决于定义	固 定， 取决 于定 义的子列 数量	struct.子 列名通过 子列名取 出子列值	暂无

create table myhive.test_array(name string, work_locations array<string>)
collection items terminated by ','; 表示集合元素的分隔符是逗号

基本查询

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING where_condition]
[ORDER BY col_list]
[CLUSTER BY col_list
 | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
[LIMIT number]
```

all/distinct 全部/去重

RLIKE

+	一次或多次匹配前面的字符	zo+匹配zo或zoo
?	零次或一次匹配前面的字符	zo?匹配z或zo
p{n}	n 是非负整数。正好匹配 n 次	o{2}匹配food中的两个o
p{n,}	n 是非负整数。至少匹配 n 次	o{2}匹配foood中的所有o
p{n,m}	M 和 n 是非负整数, 其中 $n \leq m$ 。匹配至少 n 次, 至多 m 次	o{1,3}匹配foooood中的三个o
\p{P}	一个标点字符 !"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~	J\p{P}a匹配J?a
\b	匹配一个字边界	va\b匹配java中的va, 但不匹配javar中的va

\B	非字边界匹配	va\B匹配java中的va，但不匹配java中的va
\d	数字字符匹配	1[\d]匹配13
\D	非数字字符匹配	[\D]java匹配Jjava
\w	单词字符	java匹配[\w]ava
\W	非单词字符	\$java匹配[\W]java
\s	空白字符	Java 2匹配Java\s2
\S	非空白字符	java匹配j[\S]va
\f	匹配换页符	等效于\x0c和\cL
\n	匹配换行符	等效于\x0a和\cJ

Sampling采样

```

# 基于随机分桶抽样
-- 其它条件不变的话，每一次运行结果一致
select * FROM itheima.orders
  tablesample(bucket 3 out of 10 on username);

-- 完全随机，每一次运行结果不同
select * from itheima.orders
  tablesample(bucket 3 out of 10 on rand());

# 基于数据块抽样
从前往后顺序取
--取100条
select * from itheima.orders
  tablesample(100 rows);

-- 取1%数据
select * from itheima.orders
  tablesample(1 percent);

-- 取 1KB数据
select * from itheima.orders
  tablesample(1K|M|G);

```

10表示 将表数据随机划分成y份
3表示 从y里面随机抽取x份数据
col name表示基于某个列的值
rand()表示随机的基于整行

虚拟列

虚拟列是Hive内置的可以在查询语句中使用的特殊标记，可以查询数据本身的详细参数。

Hive目前可用3个虚拟列：

- INPUT__FILE__NAME，显示数据行所在的具体文件
 - BLOCK__OFFSET__INSIDE__FILE，显示数据行所在文件的偏移量
 - ROW__OFFSET__INSIDE__BLOCK，显示数据所在HDFS块的偏移量
- 此虚拟列需要设置：SET hive.exec.rowoffset=true 才可使用

-- 类型转换函数

-- 转二进制

```
select binary('hadoop');
```

-- 自由转换，类型转换失败报错或返回NULL

```
select cast('1' as bigint);
```

-- 日期函数

-- 当前时间戳

```
select current_timestamp();
```

-- 当前日期

```
select current_date();
```

-- 时间戳转日期

```
select to_date(current_timestamp());
```

-- 年月日季度等

```
select year('2020-01-11');
```

```
select month('2020-01-11');
```

```
select day('2020-01-11');
```

```
select quarter('2020-05-11');
```

```
select dayofmonth('2020-05-11');
```

```
select hour('2020-05-11 10:36:59');
```

```
select minute('2020-05-11 10:36:59');
```

```
select second('2020-05-11 10:36:59');
```

```
select weekofyear('2020-05-11 10:36:59');
```

-- 日期之间的天数

```
select datediff('2022-12-31', '2019-12-31');
```

-- 日期相加、相减

```
select date_add('2022-12-31', 5);
```

```
select date_sub('2022-12-31', 5);
```

社交案例操作SQL

准备数据

-- 创建数据库

```
create database db_msg;
```

-- 选择数据库

```
use db_msg;
```

-- 如果表已存在就删除

```
drop table if exists db_msg.tb_msg_source ;
```

-- 建表

```
create table db_msg.tb_msg_source(  
    msg_time string comment "消息发送时间",  
    sender_name string comment "发送人昵称",  
    sender_account string comment "发送人账号",  
    sender_sex string comment "发送人性别",
```

```

sender_ip string comment "发送人ip地址",
sender_os string comment "发送人操作系统",
sender_phonetype string comment "发送人手机型号",
sender_network string comment "发送人网络类型",
sender_gps string comment "发送人的GPS定位",
receiver_name string comment "接收人昵称",
receiver_ip string comment "接收人IP",
receiver_account string comment "接收人账号",
receiver_os string comment "接收人操作系统",
receiver_phonetype string comment "接收人手机型号",
receiver_network string comment "接收人网络类型",
receiver_gps string comment "接收人的GPS定位",
receiver_sex string comment "接收人性别",
msg_type string comment "消息类型",
distance string comment "双方距离",
message string comment "消息内容"
);

-- 上传数据到HDFS(Linux命令)
hadoop fs -mkdir -p /chatdemo/data
hadoop fs -put chat_data-30w.csv /chatdemo/data/

-- 加载数据到表中, 基于HDFS加载
load data inpath '/chatdemo/data/chat_data-30w.csv' into table tb_msg_source;

-- 验证数据加载
select * from tb_msg_source tablesample(100 rows);
-- 验证一下表的数量
select count(*) from tb_msg_source;

```

ETL清洗转换

```

create table db_msg.tb_msg_etl(
    msg_time string comment "消息发送时间",
    sender_name string comment "发送人昵称",
    sender_account string comment "发送人账号",
    sender_sex string comment "发送人性别",
    sender_ip string comment "发送人ip地址",
    sender_os string comment "发送人操作系统",
    sender_phonetype string comment "发送人手机型号",
    sender_network string comment "发送人网络类型",
    sender_gps string comment "发送人的GPS定位",
    receiver_name string comment "接收人昵称",
    receiver_ip string comment "接收人IP",
    receiver_account string comment "接收人账号",
    receiver_os string comment "接收人操作系统",
    receiver_phonetype string comment "接收人手机型号",
    receiver_network string comment "接收人网络类型",
    receiver_gps string comment "接收人的GPS定位",
    receiver_sex string comment "接收人性别",

```

```

msg_type string comment "消息类型",
distance string comment "双方距离",
message string comment "消息内容",
msg_day string comment "消息日",
msg_hour string comment "消息小时",
sender_lng double comment "经度",
sender_lat double comment "纬度"
);

INSERT OVERWRITE TABLE db_msg.tb_msg_etl
SELECT
    *,
    DATE(msg_time) AS msg_day,
    HOUR(msg_time) AS msg_hour,
    SPLIT(sender_gps, ',')[0] AS sender_lng,
    SPLIT(sender_gps, ',')[1] AS sender_lat
FROM db_msg.tb_msg_source
WHERE LENGTH(sender_gps) > 0;

```

指标计算

需求1

```

--保存结果表
CREATE TABLE IF NOT EXISTS tb_rs_total_msg_cnt
COMMENT "每日消息总量" AS
SELECT
    msg_day,
    COUNT(*) AS total_msg_cnt
FROM db_msg.tb_msg_etl
GROUP BY msg_day;

```

需求2

```

--保存结果表
CREATE TABLE IF NOT EXISTS tb_rs_hour_msg_cnt
COMMENT "每小时消息量趋势" AS
SELECT
    msg_hour,
    COUNT(*) AS total_msg_cnt,
    COUNT(DISTINCT sender_account) AS sender_user_cnt,
    COUNT(DISTINCT receiver_account) AS receiver_user_cnt
FROM db_msg.tb_msg_etl GROUP BY msg_hour;

```

需求3

```

CREATE TABLE IF NOT EXISTS tb_rs_loc_cnt
COMMENT '今日各地区发送消息总量' AS
SELECT

```

```

msg_day,
sender_lng,
sender_lat,
COUNT(*) AS total_msg_cnt
FROM db_msg.tb_msg_etl
GROUP BY msg_day, sender_lng, sender_lat;

```

需求4

```

--保存结果表
CREATE TABLE IF NOT EXISTS tb_rs_user_cnt
COMMENT "今日发送消息人数、接受消息人数" AS
SELECT
msg_day,
COUNT(DISTINCT sender_account) AS sender_user_cnt,
COUNT(DISTINCT receiver_account) AS receiver_user_cnt
FROM db_msg.tb_msg_etl
GROUP BY msg_day;

```

需求5

```

--保存结果表
CREATE TABLE IF NOT EXISTS db_msg.tb_rs_s_user_top10
COMMENT "发送消息条数最多的Top10用户" AS
SELECT
sender_name AS username,
COUNT(*) AS sender_msg_cnt
FROM db_msg.tb_msg_etl
GROUP BY sender_name
ORDER BY sender_msg_cnt DESC
LIMIT 10;

```

需求6

```

CREATE TABLE IF NOT EXISTS db_msg.tb_rs_r_user_top10
COMMENT "接收消息条数最多的Top10用户" AS
SELECT
receiver_name AS username,
COUNT(*) AS receiver_msg_cnt
FROM db_msg.tb_msg_etl
GROUP BY receiver_name
ORDER BY receiver_msg_cnt DESC
LIMIT 10;

```

需求7

```

CREATE TABLE IF NOT EXISTS db_msg.tb_rs_sender_phone
COMMENT "发送人的手机型号分布" AS
SELECT
sender_phonetype,
COUNT(sender_account) AS cnt
FROM db_msg.tb_msg_etl
GROUP BY sender_phonetype;

```

需求8

--保存结果表

```
CREATE TABLE IF NOT EXISTS db_msg.tb_rs_sender_os
```

```
COMMENT "发送人的OS分布" AS
```

```
SELECT
```

```
    sender_os,
```

```
    COUNT(sender_account) AS cnt
```

```
FROM db_msg.tb_msg_etl
```

```
GROUP BY sender_os
```

Hive列注释、表注释等乱码解决方案

-- 在Hive的MySQL元数据库中执行

```
use hive;
```

1). 修改字段注释字符集

```
alter table COLUMNS_V2 modify column COMMENT varchar(256) character set utf8;
```

2). 修改表注释字符集

```
alter table TABLE_PARAMS modify column PARAM_VALUE varchar(4000) character set utf8;
```

3). 修改分区表参数，以支持分区键能够用中文表示

```
alter table PARTITION_PARAMS modify column PARAM_VALUE varchar(4000) character set utf8;
```

```
alter table PARTITION_KEYS modify column PKEY_COMMENT varchar(4000) character set utf8;
```

4). 修改索引注解

```
mysql> alter table INDEX_PARAMS modify column PARAM_VALUE varchar(4000) character set utf8;
```