> Hive 是基于Hadoop的一个数据仓库工具，能将结构化的数据文件映射为一张数据库表，并提供SQL查询功能，能将SQL语句转变成MapReduce任务来执行。
>
> 简单来说，可以将SQL翻译成MapReduce去Hadoop上执行，这样就使得数据开发和分析人员很方便的使用SQL来完成海量数据的统计和分析，而不必使用编程语言开发MapReduce那么麻烦。

Hive安装

安装Hive

将Hive上传至新建的目录 `/opt/software/` 目录下，并解压至 `/usr/local/src/` 目录下。

配置环境变量

全局生效： `/etc/profile.d/hive.sh`

只针对root账户生效： `/root/.bash_profile`

```
export HIVE_HOME=/usr/local/src/hive
export PATH=$PATH:$HIVE_HOME/bin
```

配置完成后，重启系统或使用source命令重新加载文件使其生效：

```
source /etc/profile
或
source /root/.bash_profile
```

Hive配置

修改配置文件

目录：

```
$Hive_HOME/conf/
```

在该目录下，复制 `hive-default.xml.template` 并改名为 `hive-site.xml`

```
cp hive-default.xml.template hive-site.xml
```

编辑并修改 `hive-site.xml` 的默认配置。

1. 设置jdbc连接地址：

```
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://192.168.6.86:3306/myhive?
createDatabaseIfNotExist=true&amp;characterEncoding=UTF-8</value>
```

- createDatabaseIfNotExist=true：如果指定数据库不存在，则自动创建
- `&` ：表示 符号&

- characterEncoding=UTF-8：设置字符集为UTF-8

2. 设置mysql连接用户名：

```
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
```

3. 设置mysql连接密码：

```
<name>javax.jdo.option.ConnectionPassword</name>
<value>123456</value>
```

4. 修改hive的默认使用驱动：

```
<name>javax.jdo.option.ConnectionDriverName</name><value>com.mysql.jdbc.Driver</value>
```

5. 修改临时文件目录

在配置文件中**搜索**所有 value 值包括 **system:java.io.tmpdir** 的属性项，将这些属性项的 value 值修改为绝对路径 `/opt/data/hive/tmp` （或是任意目录）；或者将这些配置项都删除。

```
<name>hive.querylog.location</name>
<name>hive.exec.local.scratchdir</name>
<name>hive.downloaded.resources.dir</name>
```

6. 可选项，显示数据库名以及列名

```
<property>
    <name>hive.cli.print.current.db</name>
    <value>true</value>
</property>

<property>
    <name>hive.cli.print.header</name>
    <value>true</value>
</property>
```

拷贝Mysql驱动包

将JDBC驱动包拷贝至 `$HIVE_HOME/lib/` 下。

初始化Hive元数据仓库

```
schematool -dbType mysql -initSchema
```

初始化成功后会提示：

```
schemaTool completed
```

Hive使用

启动Hive

```
hive
```

退出Hive

```
quit;
```

常用命令

显示所有数据库：

```
show databases;
```

新建数据库：

```
create database [数据库名];
```

删除数据库：

```
drop database [数据库名];
```

强制删除数据库：

```
drop database [数据库名] cascade;
```

查看表的字段格式：

```
desc [表名];
```

查看表的详细信息：

```
desc formatted [表名]
```

新建表：

```
create table tb_xxx[表名](id int,name string,xxx 类型) row format delimited fields
terminated by ',' lines terminated by '\n';
```

- fields terminated by：设置字段分隔符
- lines terminated by：设置行分隔符

删除表：

```
drop table [表名];
```

永久性删除表不再恢复：

```
drop table [表名] purge;
```

清空表中的数据，保留表结构：

```
truncate table [表名];
```

修改表结构：

```
-- 新增字段
alter table `tb_name` add columns (<字段名> <字段类型>)
-- 修改字段名或结构
alter table `tb_name` change `col_name` <新字段名> <新字段类型>
```

加载数据到Hive表

从本地文件加载：

```
load data local inpath '[文件目录]' overwrite into table [目标表名];
```

从HDFS中文件加载（只需去掉**local**关键字）：

```
load data inpath '[文件目录]' overwrite into table [目标表名];
```

外部表

概述

- 外部表(External table)中的数据不是Hive拥有或管理的，只管理表元数据的生命周期。
- 要创建一个外部表，需要使用 `EXTERNAL` 语法关键字。
- 删除外部表只会删除元数据，而不会删除实际数据。在Hive外部仍然可以访问实际数据。
- 实际场景中，外部表搭配 `location` 语法指定数据的路径，可以让数据更安全。

视图

Union 联合查询

UNION用于将来自于多个SELECT语句的结果合并为一个结果集。

可选参数为 DISTINCT 和 ALL（表示去重和保留所有），如果不选择默认为 DISTINCT。

注意：每个 select 语句返回的列的数量和名称必须相同。

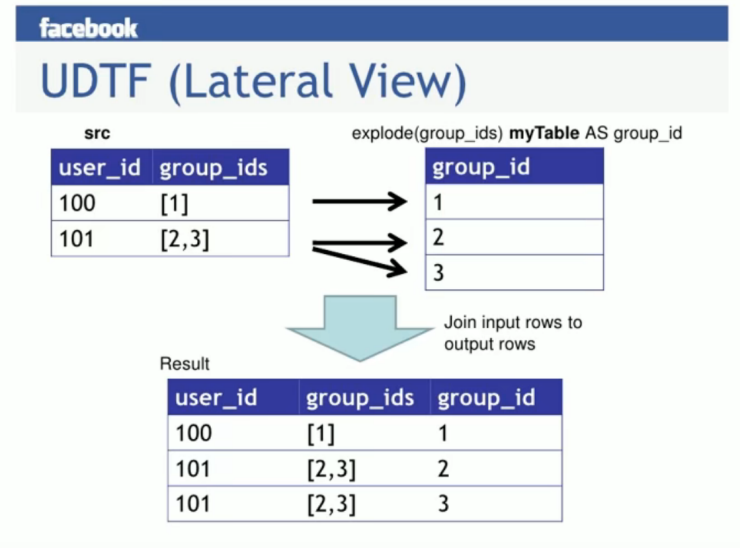
语法

```
select col1,col2 from `table1`
UNION [DISTINCT | ALL]
select clo1,clo2 from `table2`;
```

Lateral View 侧视图

原理

- 将 UDTF 的结果构建成一个类似于视图的表,然后将原表中的每一行和 UDTF 函数输出的每一行进行连接,生成一张新的虚拟表。这样就避免了UDTF的使用限制问题。
- 使用 lateral view 时也可以对 UDTF 产生的记录设置字段名称，产生的字段可以用于group by、order by、limit等语句中，不需要再单独嵌套一层子查询。



语法

```
select ... from `tableA` lateral view UDTF(xxx) as `表别名` as `col1`,`col2` ...;
```

数据类型

时间函数

取得当前日期时间：

```
-- 取得当前日期：
select current_date();
输出：2021-08-14

-- 取得当前日期时间：
select current_timestamp();
输出：2021-08-14 13:14:57

-- hive取得当前时间戳：
select unix_timestamp();
输出：1628911641
```

日期时间转日期函数

`to_date()`

```
select to_date('2021-08-14 13:34:12');
输出：2021-08-14
```

数据类型整理

Hive数据类型	对应Java数据类型	长度(数据取值范围)
TINYINT	byte	1 byte有符号(即最高位为"符号位"，下同)整数(取值范围为：-128~127)
SMALINT	short	2 byte有符号整数(取值范围为: -32768~32767)
INT	int	4 byte有符号整数(取值范围为: -2147483648~2147483647)
BIGINT	long	8 byte有符号整数(取值范围为: -9223372036854775808~9223372036854775807)
BOOLEAN	boolean	布尔类型(取值范围：true，false)
FLOAT	float	4 byte的单精度浮点数(1位符号位，8位指数，23位小数)
DOUBLE	double	8 byte的双精度浮点数(1位符号位，11位指数，52位小数)
STRING	string	可以使用单引号('')或者双引号("")定义字符串，他是一个字符集合。
TIMESTAMP		时间类型
BINARY		字节数组
DATE		日期类型

分区操作

Hive中为了提升查询操作的效率，将一个或多个字段进行分区（实际是按照该字段值保存为不同目录)，创建Hive表时指定分区列。

分区关键字：`partitioned by (分区字段名 分区字段类型)`

创建静态分区

静态分区的值是固定的。

创建动态分区

动态分区的值会按照指定字段的内容进行划分。

```
#按照操作日期分区
create table tb_class(cid int cname string,specialty string,school string) partitioned
by (etldate string) row format delimited fields terminated by ',' lines terminated by
'\n' ; ;

#按照生日日期分区，格式为yyyMM,即年月
create table tb student(sid int,sname string,sex int,birthday date,phone
string,address string,scid int,reg datedate) partitioned by (birth_date string) row
format delimited fields terminated by ',' lines terminated by '\n' stored as textfile;
```

查看分区

```
show partition 表名;
```

删除分区

```
alter table 表名 drop partition(分区字段名='分区字段值')
-- 或者是> < <= >= 等关系运算符
```

报错案例

Mysql不允许远程访问

Hive元数据初始化时报错：

```
Underlying cause: java.sql.SQLException : null,  message from server: "Host 'NMSL-666'
is not allowed to connect to this MySQL server"
```

解决方案：

打开Mysql远程连接权限：

```
use `mysql`;
update `user` set host = '%' where user = 'root';
```

刷新权限：

```
flush privileges
```

或者重启Mysql服务（以下命令为在Linux环境下）：

```
services mysqld stop
services mysqld start
```

（以下命令在Windows环境下）

```
net stop mysql
net start mysql
```

关闭版本检测

如果运行Spark代码时报错：

```
Caused by: MetaException(message:Hive Schema version 1.2.0 does not match metastore's
schema version 2.3.0 Metastore is not upgraded or corrupt)
```

```
<property>
    <name>hive.metastore.schema.verification</name>
    <value>false</value>
</property>
```