Hive 基础

刘磊

2020 年 11 月

目录

1. Hive 安装

1. 复制 apache-hive-2.3.5-bin.tar.gz 到/apps 目录下并解压。

```
cp ~/big_data_tools/apache-hive-2.3.5-bin.tar.gz /apps
tar zxvf /apps/apache-hive-2.3.5-bin.tar.gz
```

将/apps/apache-hive-2.3.5-bin，重命名为 hive。

```
mv /apps/apache-hive-2.3.5-bin/ /apps/hive
```

删除压缩包

```
rm /apps/apache-hive-2.3.5-bin.tar.gz
```

2. 设置环境变量。

```
vim ~/.bashrc
```

将 Hive 的 bin 目录，添加到用户环境变量 PATH 中

```
# Hive
export HIVE_HOME=/apps/hive
export PATH=$HIVE_HOME/bin:$PATH
```

下图是到目前位置已经设置的环境变量

```
# Java
export JAVA_HOME=/apps/java
export PATH=$JAVA_HOME/bin:$PATH

# Hadoop
export HADOOP_HOME=/apps/hadoop
export PATH=$HADOOP_HOME/bin:$PATH

# Hbase
export HBASE_HOME=/apps/hbase
export PATH=$HBASE_HOME/bin:$PATH

# Hive
export HIVE_HOME=/apps/hive
export PATH=$HIVE_HOME/bin:$PATH
```

执行 source 命令，使 Hive 环境变量生效。

```
source ~/.bashrc
```

3. 拷贝 MySQL 驱动包。由于 Hive 需要将元数据存储到 Mysql 中，所以需要拷贝 MySQL 驱动包 mysql-connector-java-5.1.46-bin.jar 到 hive 的 lib 目录下。

```
cp ~/big_data_tools/mysql-connector-java-5.1.46-bin.jar
/apps/hive/lib/
```

4. 设置配置文件

修改 hive-site.xml 文件

进入配置文件目录，将 hive-default.xml.template 重命为 hive-site.xml

```
cd /apps/hive/conf
mv hive-default.xml.template hive-site.xml
```

创建 tmp 临时目录以存储临时文件

```
mkdir -p /data/tmp/hive/tmp
```

用 vim 打开 hive-site,xml，在命令行模式执行（输入冒号:，在冒号后输入以下命令，回车）

```
%s/${system:java.io.tmpdir}/\/data\/tmp\/hive\/tmp/g
```

将${system:java.io.tmpdir}替换为上面创建的临时目录/data/tmp/hive/tmp，

```
  <!-- WARNING!!! You must make your changes in hive-site.xml instead.         -
->
:%s/${system:java.io.tmpdir}/\/data\/tmp\/hive\/tmp/g
```

一共有四处被替换。

```
     For this to work, hive.server2.logging.operation.enabled should be set to
true.
4 substitutions on 4 lines                                3976,5        66%
```

再在命令行模型执行

```
%s/${system:user.name}/root/g
```

将${system:user.name}替换为 root，共三处。

```
     For this to work, hive.server2.logging.operation.enabled should be set to
true.
3 substitutions on 3 lines                                3976,5        66%
```

由于 Hive 的元数据会存储在 Mysql 数据库中，所以需要在 Hive 的配置文件中，指定

Mysql 的相关信息。

在 545 行，将 javax.jdo.option.ConnectionURL 的值设置为数据库链接字符串。

```
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotEx
sit=true;characterEncoding=latin1&amp;useSSL=false</value>
    </property>
```

```
543   <property>
544     <name>javax.jdo.option.ConnectionURL</name>
545     <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExsit=true;c
    haracterEncoding=latin1&amp;useSSL=false</value>
546     <description>
547       JDBC connect string for a JDBC metastore.
548       To use SSL to encrypt/authenticate the connection, provide database-s
    pecific SSL flag in the connection URL.
549       For example, jdbc:postgresql://myhost/db?ssl=true for postgres databa
    se.
550     </description>
551   </property>
```

在 1020 行修改 javax.jdo.option.ConnectionDriverName 的值为连接数据库的驱动包。

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

```
1018   <property>
1019     <name>javax.jdo.option.ConnectionDriverName</name>
1020     <value>com.mysql.jdbc.Driver</value>
1021     <description>Driver class name for a JDBC metastore</description>
1022   </property>
```

在 1045 行修改 javax.jdo.option.ConnectionUserName 的值为 Mysql 数据库用户名为 root。

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

```
1044     <name>javax.jdo.option.ConnectionUserName</name>
1045     <value>root</value>
1046     <description>Username to use against metastore database</description>
1047   </property>
1048   <property>
```

在 530 行修改 javax.jdo.option.ConnectionPassword 的值为 Mysql 数据库的密码为 123456。Mysql 账号和密码安装 Mysql 的时候设置。

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

```
528   <property>
529     <name>javax.jdo.option.ConnectionPassword</name>
530     <value>123456</value>
531     <description>password to use against metastore database</description>
532   </property>
```

修改 hive-env.sh 文件

首先我们将 hive-env.sh.template 重命名为 hive-env.sh。

```
mv /apps/hive/conf/hive-env.sh.template /apps/hive/conf/hive-env.sh
```

去掉 48 行的注释，修改为 Hadoop 的安装目录

```
HADOOP_HOME=/apps/hadoop
```

去掉 51 行的注释，修改为 Hive 的配置文件目录

```
export HIVE_CONF_DIR=/apps/hive/conf
```

设置日志文件保持位置

首先我们将 hive-log4j2.properties.template 重命名为 hive-log4j2.properties。

```
mv hive-log4j2.properties.template hive-log4j2.properties
```

将 24 行修改为

```
property.hive.log.dir = /data/tmp/hive/tmp/log
```

```
21 # list of properties
22 property.hive.log.level = INFO
23 property.hive.root.logger = DRFA
24 property.hive.log.dir = /data/tmp/hive/tmp/log
25 property.hive.log.file = hive.log
```

5. 安装配置 Mysql，用于存储 Hive 的元数据。

使用 apt 命令安装 MySQL

```
sudo apt install mysql-server
```

安装以后服务会自动启动，也可执行以下命令，查看 Mysql 的运行状态。

```
sudo service mysql status
```

```
lei@ubuntu:/apps/hive/conf$ sudo service mysql status
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en
   Active: active (running) since Mon 2020-08-31 17:11:21 CST; 57s ago
 Main PID: 10409 (mysqld)
    Tasks: 27 (limit: 4667)
   CGroup: /system.slice/mysql.service
           └─10409 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pi

8月  31 17:11:20 ubuntu systemd[1]: Starting MySQL Community Server...
8月  31 17:11:21 ubuntu systemd[1]: Started MySQL Community Server.
```

通过输出，可以看出 Mysql 已启动。如果没有启动可以执行一下命令进行启动

```
sudo service mysql start
```

为 root 用户添加密码

打开终端，切换到 root 用户

```
sudo -i
```

运行 mysql 进入 Mysql 交互模式

```
mysql
```

```
lei@ubuntu:/apps/hive/conf$ sudo -i
root@ubuntu:~# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.31-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

执行以下命令将 root 密码设置为 123456。注意下面三行是一条命令。

```
update mysql.user set
authentication_string=PASSWORD('123456'),plugin='mysql_native_passwor
d' where user='root';
```

运行以下命令使修改生效

```
flush privileges;
```

执行 exit,退出 MySQL。

经过上面的设置，以后进入 MySQL 就不需要切换到系统 root 用户了，使用如下命令进行登录

```
mysql -u root -p
```

此时会提示输入密码，输入上面设置的密码 123456，回车。

```
lei@ubuntu:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.31-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

6. 创建名为 hive 的数据库，编码格式为 latin1，用于存储元数据。在 Mysql 交互模式下，执行以下命令创建名为 hive 的数据库

```
create database hive CHARACTER SET latin1;
```

查看数据库是否创建成功。

```
show databases;
```

```
mysql> create database hive CHARACTER SET latin1;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hive               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

7. 远程登录设置

我们在后面的章节会从 Hive 往 MySQL 导数据，需要从其他节点访问 MySQL，因此要授予登录用户远程连接的权限，这里以 root 账号为例，执行以下命令

```
grant all privileges on *.* to root@'%' identified by '123456';
flush privileges;
```

执行以后就会在库 Mysql 的 表 user 中添加相应的信息。输入以下命令进行查看

```
select user,Host from mysql.user;
```

```
mysql> select user, Host from mysql.user;
+------------------+-----------+
| user             | Host      |
+------------------+-----------+
| root             | %         |
| debian-sys-maint | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
5 rows in set (0.00 sec)
```

另外，需要在 MySQL 配置文件/etc/mysql/mysql.conf.d/mysqld.cnf 中注释掉 43 行

```
40 #
41 # Instead of skip-networking the default is now to listen only on
42 # localhost which is more compatible and is not less secure.
43 #bind-address      = 127.0.0.1
44 #
```

设置以后重启 MySQL 服务

```
sudo service mysql restart
```

8. 初始化元数据库。执行以下命令初始化元数据库

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

```
lei@ubuntu:~$ schematool -dbType mysql -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:mysql://localhost:3306/hive?createDatabase
IfNotExsit=true;characterEncoding=latin1&useSSL=false
Metastore Connection Driver :    com.mysql.jdbc.Driver
Metastore connection User:       root
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.mysql.sql
Initialization script completed
schemaTool completed
```

9. 启动 Hive。初始化以后，就可以启动 Hive 了。由于 Hive 对数据的处理，依赖 MapReduce 计算模型，所以启动 Hive 前需要保证 Hadoop 相关进程已经启动。在终端命令行界面，直接输入 hive 便可启动 Hive 命令行模式。

```
hive
```

```
lei@ubuntu:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/apps/hive/lib/hive-common-2
.3.5.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive>
```

输入 Hive QL 语句查询数据库，测试 Hive 是否可以正常使用。比如，显示所有数据库，

我们可以看到 Hive 有一个默认的库 default。

```
show databases;
```

```
hive> show databases;
OK
default
Time taken: 0.067 seconds, Fetched: 1 row(s)
```

Hive 安装完成。

2. Hive QL

Hive QL 是 Hive 支持的类 SQL 的查询语言。Hive QL 大体可以分为 DDL，DML，UDF 三种类型。DDL(Data Definition Language) 可以创建数据库，数据表，进行数据库和表的删除；DML(Data Manipulation Language) 可以进行数据的添加、查询；UDF(User Defined Function) 支持用户自定义查询函数。

2.1 Hive 中的数据模型

Database：在 HDFS 中表现为${hive.metastore.warehouse.dir}目录下一个文件夹

```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/user/hive/warehouse</value>
  <description>location of default database for the warehouse</description>
</property>
```

Table：在 HDFS 中表现为所属 database 目录下一个文件夹

External table：与 Table 类似，不过其数据存放位置可以指定任意 HDFS 目录路径

Partition：在 HDFS 中表现为 table 目录下的子目录

Bucket：在 HDFS 中表现为同一个表目录或者分区目录下根据某个字段的值进行 hash 散列之后的多个文件

View：与传统数据库类似，只读，基于基本表创建

2.2DDL 数据库定义语言

2.2.1 内部表

与数据库中的 Table 在概念上是类似，每一个 Table 在 Hive 中都有一个相应的目录存储数据。例如，一个表 test，它在 HDFS 中的路径为：warehouse/test。 warehouse 是在 hive-site.xml 中由 ${hive.metastore.warehouse.dir} 指定的数据仓库的目录

```
create table iris(Index int, SepalLength float, SepalWidth float,
PetalLength float, PetalWidth float, Species string) row format
delimited fields terminated by ',' stored as textfile;
```

```
hive> create table iris(Index int, SepalLength float, SepalWidth float, PetalLen
gth float, PetalWidth float, Species string) row format delimited fields termina
ted by ',' stored as textfile;
```

加载数据

Load 操作只是单纯的复制/移动操作，将数据文件移动到 Hive 表对应的位置。如果表中存在分区，则必须指定分区名。

从本地加载数据

加载本地数据，指定 LOCAL 关键字，即本地，可以同时给定分区信息 。

```
load data local inpath '/home/lei/iris.csv' into table iris;
```

```
hive> load data local inpath '/home/lei/iris.csv' into table iris;
Loading data to table default.iris
OK
Time taken: 1.026 seconds
```

从 HDFS 中加载

Hive 会使用在 Hadoop 配置文件中定义的 fs.defaultFS 指定的 Namenode 的 URI 来自动拼接完整路径。

清空表格，重新从 HDFS 加载数据

```
hive> truncate table iris;
OK
Time taken: 0.462 seconds
```

提前在 HDFS 上创建文件夹/input/iris/，并上传数据文件 iris.csv。

```
load data inpath '/input/iris/iris.csv' into table iris;
```

```
hive> load data inpath '/input/iris/iris.csv' into table iris;
Loading data to table default.iris
OK
Time taken: 1.004 seconds
```

OVERWRITE

指定 OVERWRITE，目标表（或者分区）中的内容（如果有）会被删除，然后再将路径指向的文件/目录中的内容添加到表/分区中。

```
load data inpath '/input/iris/iris.csv' overwrite into table iris;
```

2.2.2 外部表

外部表和内部表在元数据的组织上是相同的，而实际数据的存储则有较大的差异。外部表主要指向已经在 HDFS 中存在的数据，可以创建 Partition。注意：location 后面跟的是

目录，不是文件，Hive 将依据默认配置的 HDFS 路径，自动将整个目录下的文件都加载到表中。Hive 默认数据仓库路径下，不会生成外部表的文件目录。

```
create external table iris_external(Index int,SepalLength float,
SepalWidth float, PetalLength float, PetalWidth float, Species
string) row format delimited fields terminated by ',' location
'/input/iris/';
```

查看表信息

使用如下命令可以查看表的详细信息，包括 location 的指向。

```
desc formatted iris_external;
```

```
hive> desc formatted iris_external;
OK
# col_name              data_type               comment

index                   int
sepallength             float
sepalwidth              float
petallength             float
petalwidth              float
species                 string

# Detailed Table Information
Database:               default
Owner:                  lei
CreateTime:             Mon Nov 02 21:33:13 CST 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:9000/input/iris
Table Type:             EXTERNAL_TABLE
```

查看数据

```
hive> select * from iris_external;
OK
1       5.1     3.5     1.4     0.2     Iris-setosa
2       4.9     3.0     1.4     0.2     Iris-setosa
3       4.7     3.2     1.3     0.2     Iris-setosa
4       4.6     3.1     1.5     0.2     Iris-setosa
5       5.0     3.6     1.4     0.2     Iris-setosa
```

外部表与内部表的差异

1) 内部表的创建过程和数据加载过程（这两个过程可以在同一个语句中完成），在加载数据的过程中，实际数据会被移动到数据仓库目录中；之后对数据对访问将会直接在数据仓库目录中完成。删除表时，表中的数据和元数据将会被同时删除；

2) 外部表只有一个过程，加载数据和创建表同时完成，并不会移动数据到数据仓库目录中，只是与外部数据建立一个链接。当删除一个外部表时，仅删除该链接。

2.2.3 分区表

所谓分区（Partition） 对应于数据库的 Partition 列的密集索引。分区字段要写在 partitoned by（）中。

```
create table iris_partition(Index int,SepalLength float, SepalWidth float, PetalLength float, PetalWidth float) partitioned by(Species string) row format delimited fields terminated by ',' stored as textfile;
```

```
hive> create table iris_partition(Index int,SepalLength float, SepalWidth float,
 PetalLength float, PetalWidth float) partitioned by(Species string) row format
delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.221 seconds
```

加载数据

将三个类别的鸢尾花数据分别存到三个 csv 文件中

```
head -50 /home/lei/iris.csv > setosa.csv
sed -n '51,100p' /home/lei/iris.csv > versicolor.csv
sed -n '101,150p' /home/lei/iris.csv > virginica.csv
```

然后把三个文件的数据加载到分区表的三个分区中，

```
load data local inpath '/home/lei/setosa.csv' into table iris_partition partition(Species='setosa');
```

```
hive> load data local inpath '/home/lei/setosa.csv' into table iris_patition par
tition(Species='setosa');
Loading data to table default.iris_patition partition (species=setosa)
OK
Time taken: 0.687 seconds
```

```
load data local inpath '/home/lei/versicolor.csv' into table
iris_partition partition(Species='versicolor');
```

```
hive> load data local inpath '/home/lei/versicolor.csv' into table iris_patition
 partition(Species='versicolor');
Loading data to table default.iris_patition partition (species=versicolor)
OK
Time taken: 0.611 seconds
```

```
load data local inpath '/home/lei/virginica.csv' into table
iris_partition partition(Species='virginica');
```

```
hive> load data local inpath '/home/lei/virginica.csv' into table iris_patition
partition(Species='virginica');
Loading data to table default.iris_patition partition (species=virginica)
OK
Time taken: 0.642 seconds
```

增加分区

```
alter table iris_partition add partition (Species='unknown');
```

```
hive> alter table iris_partition add partition (species='unknown');
OK
Time taken: 0.141 seconds
```

查看分区

```
show partitions iris_partition;
```

```
hive> show partitions iris_partition;
OK
species=setosa
species=unknown
species=versicolor
species=virginica
Time taken: 0.115 seconds, Fetched: 4 row(s)
```

删除分区

```
alter table iris_partition drop if exists partition
(Species='unknown');
```

```
hive> alter table iris_partition drop if exists partition (species='unknown');
Dropped the partition species=unknown
OK
Time taken: 0.193 seconds
hive> show partitions iris_partition;
OK
species=setosa
species=versicolor
species=virginica
Time taken: 0.104 seconds, Fetched: 3 row(s)
```

查询分区表数据

```
select * from iris_partition;
```

```
6.7     3.3     5.7     2.5     virginica
6.7     3.0     5.2     2.3     virginica
6.3     2.5     5.0     1.9     virginica
6.5     3.0     5.2     2.0     virginica
6.2     3.4     5.4     2.3     virginica
5.9     3.0     5.1     1.8     virginica
Time taken: 0.236 seconds, Fetched: 150 row(s)
```

按分区查询

```
select * from iris_partition where species='setosa';
```

按 sepallength 排序查询前 5 条记录

```
select * from iris_partition sort by sepallength desc limit 5;
```

```
132     7.9     3.8     6.4     2.0     virginica
136     7.7     3.0     6.1     2.3     virginica
118     7.7     3.8     6.7     2.2     virginica
119     7.7     2.6     6.9     2.3     virginica
123     7.7     2.8     6.7     2.0     virginica
Time taken: 88.442 seconds, Fetched: 5 row(s)
```

查询表结构

```
desc iris_partition;
```

```
hive> desc iris_partition;
OK
sepal_length            float
sepal_width             float
petal_length            float
petal_width             float
species                 string

# Partition Information
# col_name              data_type               comment

species                 string
Time taken: 0.105 seconds, Fetched: 10 row(s)
```

在 Hive 中，表中的一个 Partition 对应于表下的一个目录，所有的 Partition 的数据都存储在对应的目录中。

```
lei@ubuntu:~$ hadoop fs -ls /user/hive/warehouse/iris_partition/
Found 3 items
drwxr-xr-x   - lei supergroup          0 2020-08-31 23:30 /user/hive/warehouse/iris_partition/species=setosa
drwxr-xr-x   - lei supergroup          0 2020-08-31 23:31 /user/hive/warehouse/iris_partition/species=versicolor
drwxr-xr-x   - lei supergroup          0 2020-08-31 23:31 /user/hive/warehouse/iris_partition/species=virginica
```

2.2.4 桶表

桶是比表或分区更为细粒度的数据范围划分。针对某一列进行桶的组织，对列值哈希，然后除以桶的个数求余，决定将该条记录存放到哪个桶中。跟 MapReduce 中的 HashPartitioner 的原理一样。在 MR 中，按照 key 的 Hash 值去模除以 reductTask 的个数。在 Hive 中时按照分桶字段的 Hash 值去模除以分桶的个数。 分桶的主要作用：数据抽样、提高查询效率。

创建桶表

```
create table iris_bucket(Index int, SepalLength float, SepalWidth float, PetalLength float, PetalWidth float, Species string) clustered by(Species) into 3 buckets row format delimited fields terminated by ',';
```

```
hive> create table iris_bucket(Index int, SepalLength float, SepalWidth float, P
etalLength float, PetalWidth float, Species string) clustered by(Species) into 3
 buckets row format delimited fields terminated by ',';
OK
Time taken: 0.205 seconds
```

说明：

- clustered by：指定需要进行分桶的列
- into x buckets：指定进行分桶的数量

加载数据

导入数据不能使用 load data 这种方式，需要从别的表来引用，使用 load data 加载数据，则不能生成桶数据。另外，需要先 set hive.enforce.bucketing = true，才可以将数据正常写入桶中。

```
insert overwrite table iris_bucket select * from iris;
```

```
hive> insert overwrite table iris_bucket select * from iris;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = lei_20200901100335_8933d161-76b3-440a-8c0b-9c8b2e1e947e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1598924288950_0002, Tracking URL = http://ubuntu:8088/proxy/a
pplication_1598924288950_0002/
Kill Command = /apps/hadoop/bin/hadoop job  -kill job_1598924288950_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 3
2020-09-01 10:03:41,348 Stage-1 map = 0%,  reduce = 0%
2020-09-01 10:03:46,493 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.73 se
c
2020-09-01 10:03:58,855 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 1.64 s
ec
2020-09-01 10:04:02,986 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 2.56 s
ec
2020-09-01 10:04:06,050 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.47
sec
MapReduce Total cumulative CPU time: 3 seconds 470 msec
Ended Job = job_1598924288950_0002
Loading data to table default.iris_bucket
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 3   Cumulative CPU: 3.47 sec   HDFS Read: 22401 H
DFS Write: 3968 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 470 msec
OK
Time taken: 31.751 seconds
```

HDFS 上的目录结构

```
lei@ubuntu:~$ hadoop fs -ls /user/hive/warehouse/iris_bucket
Found 3 items
-rwxr-xr-x   1 lei supergroup          0 2020-11-02 22:12 /user/hive/warehouse/i
ris_bucket/000000_0
-rwxr-xr-x   1 lei supergroup       1750 2020-11-02 22:12 /user/hive/warehouse/i
ris_bucket/000001_0
-rwxr-xr-x   1 lei supergroup       3292 2020-11-02 22:12 /user/hive/warehouse/i
ris_bucket/000002_0
```

查询全部数据

```
select * from iris_bucket;
```

抽样查询

```
select * from iris_bucket tablesample(bucket 1 out of 3 on Species);
```

用法：TABLESAMPLE(BUCKET x OUT OF y) 。y 必须是 Table 总 Bucket 数的倍数或者因子。Hive 根据 y 的大小，决定抽样的比例。例如，Table 总共分了 64 份，当 y=32 时，抽取(64/32=)2 个 bucket 的数据，当 y=128 时，抽取(64/128=)1/2 个 bucket 的数据。x 表示从哪个 Bucket 开始抽取。例如，Table 总 Bucket 数为 32，tablesample(bucket 3 out of 16)，表示总共抽取（32/16=）2 个 bucket 的数据，分别为第 3 个 Bucket 和第（3+16=）19 个 Bucket 的数据。

分区中的数据可以被进一步拆分成桶

```
create table iris_partition_bucket(Index int, SepalLength float,
SepalWidth float, PetalLength float, PetalWidth float) partitioned
by(Species string) clustered by(SepalLength) into 3 buckets row
format delimited fields terminated by ',';
```

```
hive> create table iris_partition_bucket(Index int, SepalLength float, SepalWidt
h float, PetalLength float, PetalWidth float) partitioned by(Species string) clu
stered by(SepalLength) into 3 buckets row format delimited fields terminated by
',';
OK
Time taken: 0.186 seconds
```

插入数据

将不同类型的鸢尾花保存到不同的分区并将数据分桶。

```
insert overwrite table iris_partition_bucket
partition(Species='setosa') select Index, SepalLength, SepalWidth,
PetalLength, PetalWidth from iris where species='Iris-setosa';
```

```
insert overwrite table iris_partition_bucket
partition(Species='versicolor') select Index, SepalLength,
SepalWidth, PetalLength, PetalWidth from iris where species='Iris-
versicolor';
```

```
insert overwrite table iris_partition_bucket
partition(Species='virginica') select Index, SepalLength, SepalWidth,
PetalLength, PetalWidth from iris where species='Iris-virginica';
```

指定 overwrite 关键字，目标文件夹中之前存在的数据将会被先删除掉，如果没有这个关键字，仅仅会把新增的文件增加到目标文件夹中，而不会删除之前的数据。

目录结构

在 HDFS 上每个分区是一个目录

```
lei@ubuntu:~$ hadoop fs -ls /user/hive/warehouse/iris_partition_bucket
Found 3 items
drwxr-xr-x   - lei supergroup          0 2020-11-02 22:38 /user/hive/warehouse/iris_partition_bucket/species=setosa
drwxr-xr-x   - lei supergroup          0 2020-11-02 22:40 /user/hive/warehouse/iris_partition_bucket/species=versicolor
drwxr-xr-x   - lei supergroup          0 2020-11-02 22:41 /user/hive/warehouse/iris_partition_bucket/species=virginica
```

每个分区目录下有分桶数个文件

```
lei@ubuntu:~$ hadoop fs -ls /user/hive/warehouse/iris_partition_bucket/species=setosa
Found 3 items
-rwxr-xr-x   1 lei supergroup        189 2020-11-02 22:38 /user/hive/warehouse/iris_partition_bucket/species=setosa/000000_0
-rwxr-xr-x   1 lei supergroup        224 2020-11-02 22:38 /user/hive/warehouse/iris_partition_bucket/species=setosa/000001_0
-rwxr-xr-x   1 lei supergroup        528 2020-11-02 22:38 /user/hive/warehouse/iris_partition_bucket/species=setosa/000002_0
```

桶的概念其实就是 MapReduce 的分区的概念，两者完全相同。物理上每个桶就是目录里的一个文件，一个作业产生的桶（输出文件）数量和 reduce 任务个数相同。

而分区表的概念，则是新的概念。分区代表了数据的仓库，也就是文件夹目录。每个文件夹下面可以放不同的数据文件。通过文件夹可以查询里面存放的文件。但文件夹本身和数据的内容毫无关系。桶则是按照数据内容的某个值进行分桶，把一个大文件散列称为一个个小文件。 这些小文件可以单独排序。如果另外一个表也按照同样的规则分成了一个个小文件。两个表 join 的时候，就不必要扫描整个表，只需要匹配相同分桶的数据即可。效率当然大大提升。同样，对数据抽样的时候，也不需要扫描整个文件。只需要对每个分区按照相同规则抽取一部分数据即可。

删除表

```
DROP TABLE [IF EXISTS] table_name;
```

DROP TABLE 语句用于删除表的数据和元数据（表结构）。对应外部表，只删除 Metastore 中的元数据，而外部数据保存不动。

如果只想删除表数据，保留表结构，跟 MYSQL 类似，使用 TRUNCATE 语句。truncate 不能删除外部表！因为外部表里的数据并不是存放在 Hive Meta store 中

```
truncate table table_name;
```

2.2.5 数据库操作

创建数据库

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

例如创建名为 hive_db_test 的数据库，并设置创建者和创建时间。

```
create database if not exists hive_db_test with dbproperties
('creator' ='lei', 'date' = '2020-08-31');
```

```
hive> show databases;
OK
default
hive_db_test
Time taken: 0.02 seconds, Fetched: 2 row(s)
```

切换数据库

```
use hive_db_test;
```

查看数据库

```
describe database hive_db_test;
```

```
hive> describe database hive_db_test;
OK
hive_db_test            hdfs://localhost:9000/user/hive/warehouse/hive_db_test.d
b       lei     USER
Time taken: 0.058 seconds, Fetched: 1 row(s)
```

可以看到数据库在 HDFS 上的存储位置 hdfs://localhost:9000/user/hive/warehouse/，

每个数据库一个文件。

查看当前使用的数据库

```
select current_database();
```

```
hive> select current_database();
OK
default
Time taken: 4.308 seconds, Fetched: 1 row(s)
```

删除数据库

```
drop database hive_db_test;
```

默认情况下，Hive 不允许用户删除一个包含表的数据库。用户要么先删除数据库中的表，

再删除数据库；要么在删除命令的最后加上关键字 CASCADE,这样 Hive 会先删除数据库

中的表，再删除数据库，

命令如下（务必谨慎使用此命令）

```
drop database my_hive_test CASCADE;
```

查看所有数据库

```
show databases;
```

```
hive> show databases;
OK
default
hive_db_test
Time taken: 0.051 seconds, Fetched: 2 row(s)
```

2.3 DML 数据操作语言

2.3.1 表操作

重命名表

```
alter table iris rename to iris_flower;
```

```
hive> alter table iris rename to iris_flower;
OK
Time taken: 0.146 seconds
hive> show tables;
OK
iris_bucket
iris_flower
iris_partition_bucket
Time taken: 0.079 seconds, Fetched: 3 row(s)
```

增加列

```
alter table iris_flower add columns(id int);
```

```
hive> alter table iris_flower add columns(id int);
OK
Time taken: 0.217 seconds
hive> desc iris_flower;
OK
index                   int
sepallength             float
sepalwidth              float
petallength             float
petalwidth              float
species                 string
id                      int
Time taken: 0.074 seconds, Fetched: 7 row(s)
```

修改列名

将列 id 改为 num

```
alter table iris_flower change id num int;
```

```
hive> alter table iris_flower change id num int;
OK
Time taken: 0.25 seconds
hive> desc iris_flower;
OK
index                   int
sepallength             float
sepalwidth              float
petallength             float
petalwidth              float
species                 string
num                     int
Time taken: 0.096 seconds, Fetched: 7 row(s)
```

创建表接收查询结果

查询 iris_flower 表中 Sepal_Length<5 的数据存入新表 iris_result

```
create table iris_result as select SepalLength, SepalWidth,
PetalLength, PetalWidth from iris_flower where SepalLength<5;
```

查看结果

```
select * from iris_result;
```

导出数据

导出到本地。将 iris_result 中的数据导出到文件夹/home/lei/result（非常重要：新建一个文件夹，因为指定的目录会被清空）执行语句后，会在本地目录的/home/lei/result 下生成一个名为 000000_0 的数据文件。导出的数据列之间的分隔符默认是^A(ascii 码是\001)。

```
insert overwrite local directory '/home/lei/result' select * from
iris_result;
```

```
4.9^A3.0^A1.4^A0.2
4.7^A3.2^A1.3^A0.2
4.6^A3.1^A1.5^A0.2
4.6^A3.4^A1.4^A0.3
4.4^A2.9^A1.4^A0.2
4.9^A3.1^A1.5^A0.1
4.8^A3.4^A1.6^A0.2
4.8^A3.0^A1.4^A0.1
4.3^A3.0^A1.1^A0.1
4.6^A3.6^A1.0^A0.2
```

导出到 HDFS。将 iris_result 中的数据导出到 HDFS 文件/output/iris/。导出路径为文件夹路径，不必指定文件名。执行语句后，会在 HDFS 目录的/hivedb 下生成一个名为 000000_0 的数据文件。

```
insert overwrite directory '/output/iris/' select * from iris_result;
```

查询全表

```
select * from iris_flower;
```

查询部分字段

```
select SepalLength, PetalLength, Species from iris_flower;
```

统计每个类型的鸢尾花数量

```
select collect_set(species)[0], count(*) from iris_flower group by
species;
```

```
Iris-setosa     50
Iris-versicolor 50
Iris-virginica  50
Time taken: 40.532 seconds, Fetched: 3 row(s)
```

注意：

- Hive 不允许直接访问非 group by 字段；
- 对于非 group by 字段，可以用 Hive 的 collect_set 函数收集这些字段，返回一个数组；
- 使用数字下标，可以直接访问数组中的元素。

统计函数

```
select max(SepalLength), min(SepalWidth), avg(PetalLength),
sum(PetalWidth) from iris_flower;
```

```
Total MapReduce CPU Time Spent: 5 seconds 460 msec
OK
7.9     2.0     3.7586666552225747      179.7999987155199
Time taken: 40.08 seconds, Fetched: 1 row(s)
```

2.4 常用命令

- 查询数据库：show databases;
- 模糊搜索表：show tables like '*name*';

- 删除数据库：drop database dbname;
- 删除数据表：drop table tablename;
- 查看表结构信息：desc table_name;
- 查看详细表结构信息： desc formatted table_name;
- 查看分区信息： show partitions table_name;