

# 第6章Hive数据仓库

---

(源自:<https://biglab.site>)

(版本:Ver1.0-20231109)

## 第6章Hive数据仓库

### 任务前言

学习目标

任务背景

软件准备

测试数据

### 任务6.1认识Hive数据仓库

任务描述

什么是Hive

了解Hive与传统数据库的对比

了解Hive系统架构

了解Hive数据模型

了解Hive工作原理

### 任务6.2访问Hive的3种方式

任务描述

设置内嵌Derby模式

在master主机上执行安装:

修改系统环境变量

运行hive

查看hive:

设置直连数据库模式

安装Mysql和配置mysql库

删除已有mysql安装新mysql

修改mysql配置

启mysql服务

查看临时密码

使用临时密码登陆:

修改mysql的root密码

配置Hive直连数据库

需要配置hive-site.xml

初始化元数据库

运行hive

设置远程模式

修改master上的配置文件hive-site.xml

再次初始化元数据库

从master分发hive到从机slave1,slave2

在从机slave1和slave2上执行

在从机slave1和slave2上修改hive-site.xml文件

在master上启动hive服务

在slave1或slave2上运行hive

在slave1或slave2创建hive\_remote数据库

在master的mysql库观察

修改数据库的字符集

### 任务6.3实现Hive表的创建与修改

Hive数据定义语言DDL的基本语法

Hive表的基本操作

### 任务6.4实现Hive表中数据的增删查改

- Hive数据操作语言DML的基本语法
- Hive表的基本数据操作
- 任务6.5掉线率top10基站统计
  - 分析基本思路
  - 任务实现
    - 数据文件准备
    - 数据操作
- 小结
- 版本历史

## 任务前言

### 学习目标

1. 了解Hive的概念。
2. 了解Hive与传统数据库的异同点。
3. 了解系统架构及数据模型
4. 熟悉Hive的安装的3种模式及配置过程
5. 掌握Hive中数据库与表的创建、修改操作方法
6. 掌握Hive表数据增删查改的操作方法

### 任务背景

随着信息技术的普及和企业信息化建设步伐的加快，企业逐步认识到建立企业范围内的统一数据存储的重要性，越来越多的企业已经建立或正着手建立企业数据仓库。企业数据仓库（Data Warehouse）有效集成了来自不同部门、不同地理位置、具有不同格式的数据，为企业管理决策提供了企业范围内的单一数据视图，从而为综合分析和科学决策奠定了坚实的基础。

电信运营商为了优化维护工作，每天都会派出维护人员对分布在全国各的基站进行检测，分析基站的工作情况。其中掉话率在移动通信网中是一项非常重要的指标，掉话率的高低在一定程度上体现了移动网通信质量的优劣。不同厂家的设备对该指标的计算方法与标准不尽相同。因此，对基站掉话率的分析显得尤为重要。

运营商有一个系统，收集了分布在各地基站的各种数据，数据字段说明如下表所示，共有5个数据字段

属性名称	属性说明
record_time	通话时间（单位：s）
imei	基站编号
cell	手机编号
drop_num	掉话秒数（单位：s）
duration	通话总秒数（单位：s）

为了提升网络质量，节约时间和人力成本，优化维护工作，分析基站的掉话率就显得尤为重要，因此，针对系统收集的各基站运行数据，运营商为更精准、有针对性的完成基站的维护工作，提出了基站掉话率统计需求

通常，运营商基站运行期间产生的文件数量都比较大，而且文件中的记录数也很多，文件大多数为文本格式。为了保证较高的处理效率与灵活性，在此选用Hive数据仓库对基站数据进行存储与处理分析。本章将详细讲解怎样通过Hive编程解决实际问题，首先介绍Hive数据仓库的系统架构、数据模式和工作原理，其次讲解访问Hive的3种方式及其搭建过程，再介绍Hive数据仓库中表的创建与修改的基本语法格式，并结合官方的示例介绍表数据的增删改查，最后通过编写Hive语句实现基站掉话率的统计分析

## 软件准备

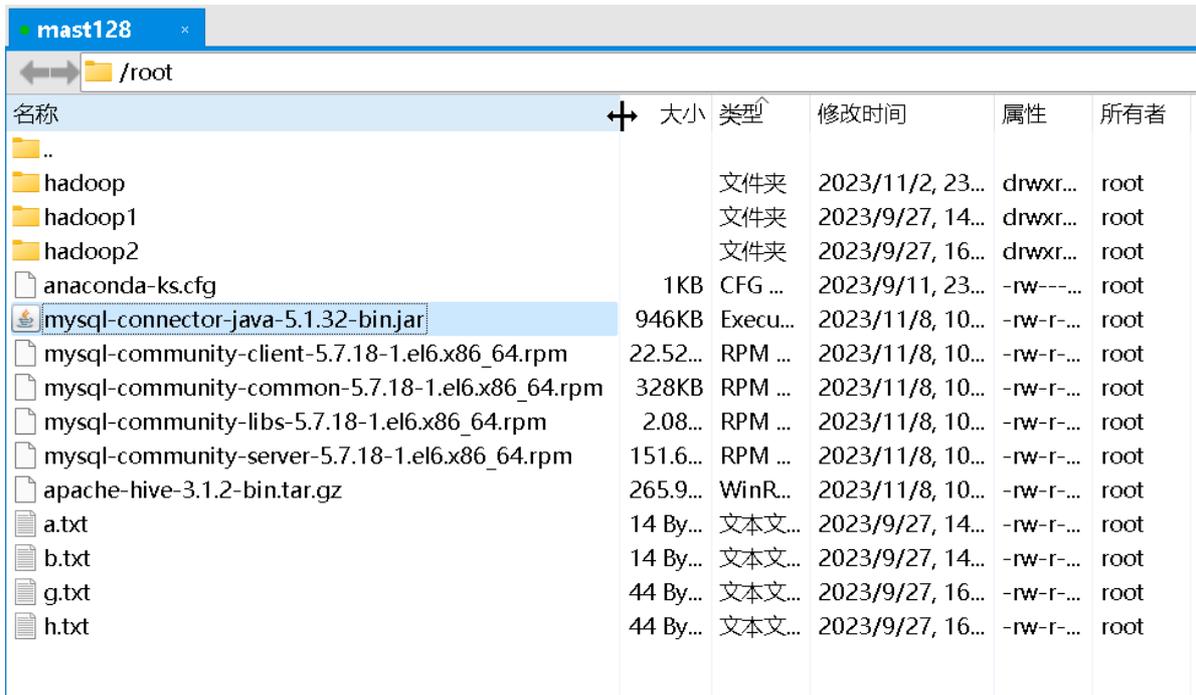
下载地址：链接：[https://pan.baidu.com/s/1DHGRnjvi4yMY41Se7\\_vDA](https://pan.baidu.com/s/1DHGRnjvi4yMY41Se7_vDA) 提取码: vbgu

软件列表：

软件	版本	安装包称
apache-hive	3.1.2	apache-hive-3.1.2-bin.tar.gz
mysql-connector-java	5.1.32	mysql-connector-java-5.1.32-bin
mysql-community-client	5.7.18	mysql-community-client-5.7.18-1.el6.x86_64.rpm
mysql-community-common	5.7.18	mysql-community-common-5.7.18-1.el6.x86_64.rpm
mysql-community-libs	5.7.18	mysql-community-libs-5.7.18-1.el6.x86_64.rpm
mysql-community-server	5.7.18	mysql-community-server-5.7.18-1.el6.x86_64.rpm

软件上传：

1. 使用xftp 将apache-hive-3.1.2-bin.tar.gz和mysql-connector-java-5.1.32-bin 上传到master虚拟机的/root目录下
2. 使用xftp 将mysql-community相关的文件上传到master虚拟机的/root目录下



## 测试数据

通过百度网盘下载：

下载地址：链接：[https://pan.baidu.com/s/1DHGRrnjvi4yMY41Se7\\_vDA](https://pan.baidu.com/s/1DHGRrnjvi4yMY41Se7_vDA) 提取码：vbgu

软件	大小	版本	安装包称
hadoop_data	99.7M	20230925	hadoop_data.tar.gz

下载hadoop\_data.tar.gz后，通过xftp工具上传到master主机的/root/hadoop目录下，然后xshell进入master执行：

```
1 cd /root/hadoop
2 tar -zxvf ./hadoop_data.tar.gz
3 hdfs dfs -mkdir /user/myname/stu
4 hdfs dfs -put ./hadoop_data/student_info.txt /user/myname/stu
5 hdfs dfs -put ./hadoop_data/students.txt /user/myname/stu
6 hdfs dfs -put ./hadoop_data/course.txt /user/myname/stu
7 hdfs dfs -put ./hadoop_data/sc.txt /user/myname/stu
8 hdfs dfs -put ./hadoop_data/jizhan_information.csv /user/myname/stu
```

## 任务6.1认识Hive数据仓库

### 任务描述

Hive数据仓库基于Hadoop开发，是Hadoop生态圈组件之一，是海量数据存储和离线批量处理的常用工具。

本小节的任务如下，这是学习与使用Hive进行数据存储与处理的第一步。

1. 了解Hive的概念。
2. 了解Hive与传统数据库的异同点。
3. 了解系统架构及数据模型。

### 什么是Hive

随着大数据时代的全面到来，传统数据仓库面临的挑战主要包括以下几个方面。

1. 无法满足快速增长的海量数据存储需求。
2. 无法有效处理不同类型的数据。
3. 计算和处理能力不足。

大多数数据仓库应用程序均是使用关系型数据库进行实现的，并使用SQL作为查询语言。为了提高数据仓库的性能，选择将基于传统关系型数据库的数据仓库构建在Hadoop上，是一个很好的解决思路。对于大量的SQL用户，包括专业数据库设计师和管理员，也包括使用SQL语句从数据仓库中抽取信息的临时用户，急需一个可以提供有效地、合理地组织和处理海量数据的模型，Hive应运而生。

Hive是Hadoop生态系统中必不可少的一个工具，它提供了一种SQL语言，可以查询存储在Hadoop分布式文件系统（Hadoop Distributed File System, HDFS）中的数据，也可以查询其他与Hadoop集成的文件系统的数据，如MapReduce-FS、Amazon S3，也可以查询数据库中的数据，如HBase和Cassandra。

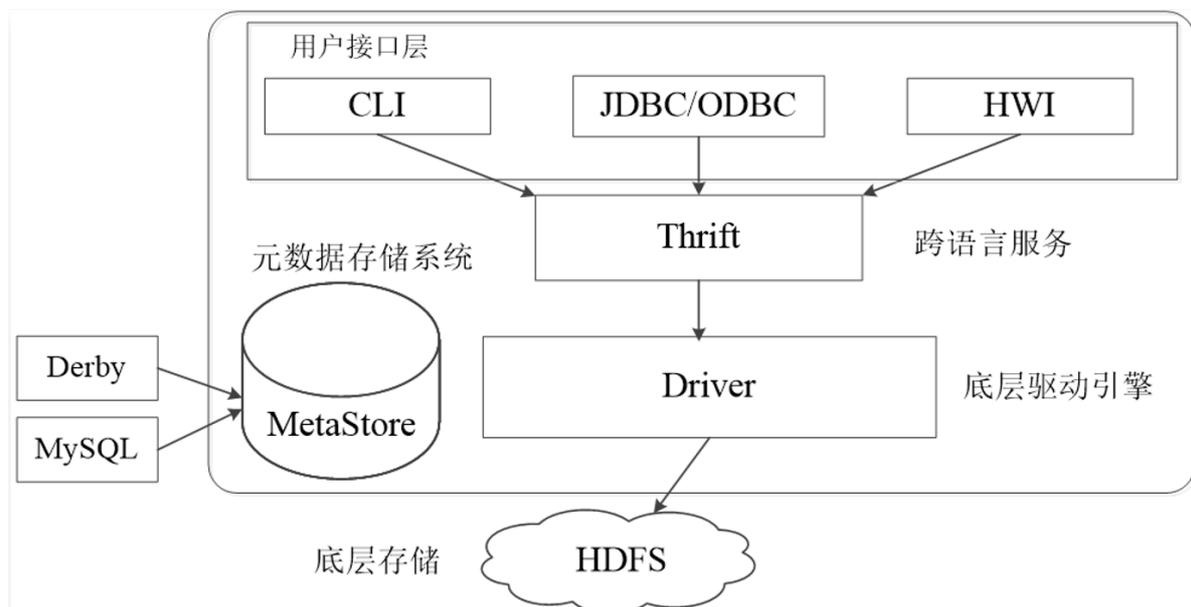
## 了解Hive与传统数据库的对比

Hive在很多方面和传统数据库类似，但是它底层依赖的是HDFS和MapReduce（或Tez、Spark），因此，Hive在很多方面又有别于传统数据库。从数据存储、索引、分区、执行引擎、执行延迟、扩展性、数据规模等方面，对Hive和传统数据库进行了对比分析，如下表所示

对比内容	Hive	传统数据库
查询语言	HQL	SQL
数据存储	HDFS	本地文件系统
索引	0.8版以后支持位图索引	支持复杂索引
执行引擎	MapReduce、Tez、Spark	自身的执行引擎
执行延迟	高	低
可扩展性	好	有限
处理数据规模	大	小

## 了解Hive系统架构

Hive是典型的客户端/服务器（Client/Server，C/S）模式，底层执行引擎使用的是Hadoop的MapReduce框架，因此Hive运行在Hadoop基础上。Hive的系统架构组成如下图所示，主要包括5个部分：用户接口层、跨语言服务、元数据存储系统、底层的驱动引擎、底层存储



Hive系统架构说明：

- 用户接口层：

用户接口用于连接访问Hive，包括CLI、JDBC/ODBC和HWI（Hive Web Interface）3种方式。CLI指的是shell终端命令行，采用交互式的方式访问Hive。JDBC/ODBC是通过客户端的方式连接并访问Hive。HWI则是通过浏览器的方式访问Hive。

- 跨语言服务

Thrift是用于进行可扩展且跨语言的服务的开发，Hive集成了Thrift服务，能让不同的编程语言调用Hive的接口

- 元数据库 (MetaStore) 存储系统

Hive将元数据存储于数据库中，如MySQL、Derby (Hive自带的内存数据库)。Hive中的元数据包括表的名字，表的列和分区及其属性，表的属性 (是否为外部表等)，表的数据所在目录等。MetaStore默认存在自带的Derby数据库中。由于Derby数据库不适合多用户操作，并且数据存储目录不固定，不方便管理，因此，通常都将元数据存储于MySQL数据库中。

- 底层的驱动引擎 (Driver)

底层的驱动引擎实现了将HiveQL语句转化为MapReduce任务的过程。Hive的底层的驱动引擎主要包括解释器、编译器、优化器和执行器，一同完成HQL查询语句从词法分析、语法分析、编译、优化以及查询计划的生成。生成的查询计划存储在HDFS中，并在随后由执行器 (Executor) 调用MapReduce执行。

- 底层存储

Hive数据仓库的数据是存储在HDFS中的。针对大部分的HQL查询请求，Hive内部会将HQL语句自动转换为MapReduce任务执行。

## 了解Hive数据模型

- 数据库

Hive数据库类似于传统数据库的Database。内部表与数据库中的table在概念上类似。每一个table在Hive中都有一个相应的目录存储数据，所有的table数据均保存在这个目录中。

- 表

Hive的表在逻辑上由存储的数据和描述表格数据形式的相关元数据组成。表存储的数据存放在分布式文件系统里，如HDFS。

Hive中的表分为两种类型，一种为内部表，这种表的数据存储在Hive数据仓库中；另一种为外部表，表数据可以存放在Hive数据仓库外的分布式文件系统中，也可以存储在Hive数据仓库中。值得一提的是，Hive数据仓库其实也是HDFS中的一个目录，该目录是Hive数据存储的默认路径，也可以在Hive的配置文件中重新配置。

- 分区和桶

Hive将表组织成分区 (partition)，这是一种根据分区列 (partition column) 的值对表进行粗略划分的机制。使用分区可以加快数据分片的查询速度。表或分区可以进一步分为桶 (bucket)。

将表或分区组织成桶有以下两个原因。

第一个原因是可以获得更高的查询处理效率。桶其实是为表加上了额外的结构，Hive在处理有些查询时能够利用这个结构。如连接两个在相同列上划分了桶的表，可以使用Map端的join高效实现。

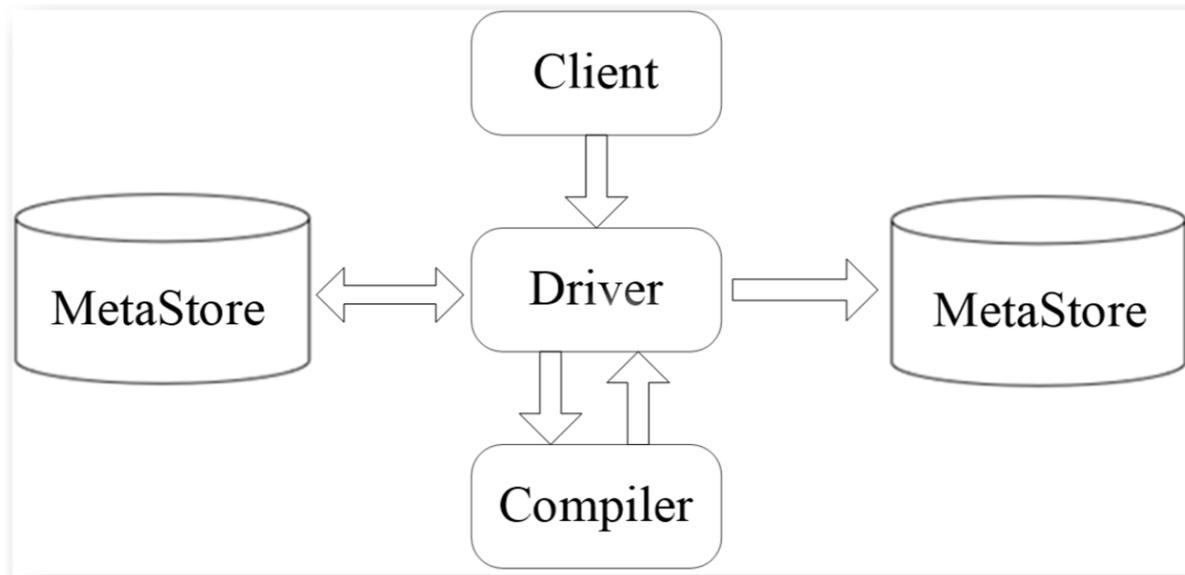
第二个原因是使取样 (sampling) 更高效。在大规模数据集上进行开发和查询时，如果可以先在一部分小数据集上运行查询，那么会更加地方便。Hive使用CLUSTERED BY子句来指定划分桶所用的列和要划分的桶的个数。

- 视图

Hive的视图与传统数据库的视图类似。视图是只读的，视图依赖的基本表如果改变，数据增加不会影响视图的呈现；如果删除基本表，那么会出现问题。

## 了解Hive工作原理

Hive的执行流程如下图所示，Hive中的任务执行流程是将客户端（Client）提交的任务（如HiveQL语句）通过驱动（Driver）找到HiveQL中的相关元数据（MetaStore）确定表及存储的情况，再编译（Compiler）相关信息，生成执行计划，提交给Hadoop完成Hive的执行过程



## 任务6.2访问Hive的3种方式

### 任务描述

Hive的安装模式分为3种，分别是内嵌模式、直连数据库模式（或本地模式）和远程模式。

1. 内嵌模式使用内嵌的Derby数据库存储元数据。
2. 直连接数据库模式采用外部数据库存储元数据。
3. 远程模式也采用外部数据库存储元数据，不同的是，远程模式需要单独开启MetaStore服务。

为了顺利地访问Hive，本小节的任务是介绍访问Hive的3种方式，并根据不同的安装模式进行配置

### 设置内嵌Derby模式

内嵌模式使用内嵌的Derby数据库存储元数据。

在内嵌模式下，只允许一个会话连接，若尝试多个会话则连接时将报错。

#### 在master主机上执行安装：

```
1 cd /root
2 tar -zxvf ./apache-hive-3.1.2-bin.tar.gz
3 mv ./apache-hive-3.1.2-bin /usr/local/hive
4 mv /usr/local/hive/lib/guava-19.0.jar /usr/local/hive/lib/guava-19.0.jar.bak
5 cp /usr/local/hadoop-3.1.4/share/hadoop/common/lib/guava-27.0-jre.jar
  /usr/local/hive/lib/guava-27.0.jar
6 mv /usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar /usr/local/hive/lib/log4j-
  slf4j-impl-2.10.0.jar.bak
7 cd /usr/local/hive/bin
8 ./schematool -dbType derby -initSchema
```

运行后会出现如下内容，说明元数据库初始化完成

```
1 Initialization script completed
2 schemaTool completed
```

## 修改系统环境变量

```
1 vi /etc/profile
```

在文件末尾添加：

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

立即生效环境变量

```
1 source /etc/profile
```

## 运行hive

(前提是hadoop集群已启动，使用jps查看，若没有，使用start-all.sh启动hadoop集群)

```
1 hive
```

运行结果如：

```
1 [root@master bin]# hive
2 SLF4J: Class path contains multiple SLF4J bindings.
3 SLF4J: Found binding in [jar:file:/usr/local/hadoop-
4 3.1.4/share/hadoop/common/lib/slf4j-log4j12-
5 1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
6 SLF4J: Found binding in [jar:file:/usr/local/hadoop-
7 3.1.4/share/hadoop/common/slf4j-log4j12-
8 1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
9 SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings_for_an
10 explanation.
11 SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
12 which: no hbase in (/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
13 3.1.4/bin:/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
14 3.1.4/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.8
15 .0_281-amd64/bin:/root/bin:/usr/java/jdk1.8.0_281-
16 amd64/bin:/bin:/usr/local/hive/bin:/usr/java/jdk1.8.0_281-
17 amd64/bin:/usr/local/hive/bin)
18 SLF4J: Class path contains multiple SLF4J bindings.
19 SLF4J: Found binding in [jar:file:/usr/local/hadoop-
20 3.1.4/share/hadoop/common/lib/slf4j-log4j12-
21 1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
22 SLF4J: Found binding in [jar:file:/usr/local/hadoop-
23 3.1.4/share/hadoop/common/slf4j-log4j12-
24 1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
```

```
11 SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an
12 explanation.
13 SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
14 Hive Session ID = ef4481eb-f38c-4f15-8d19-2028699da4c1
15 Logging initialized using configuration in
16 jar:file:/usr/local/hive/lib/hive-common-3.1.2.jar!/hive-log4j2.properties
17 Async: true
18 Hive-on-MR is deprecated in Hive 2 and may not be available in the future
19 versions. Consider using a different execution engine (i.e. spark, tez) or
20 using Hive 1.X releases.
21 Hive Session ID = a55a7daa-edef-4ce4-9358-a65a1aa457b1
22 hive>
```

## 查看hive:

```
1 hive> show databases;
2 OK
3 default
4 Time taken: 0.483 seconds, Fetched: 1 row(s)
5 hive>
6 hive> exit;
7 [root@master bin]#
```

说明数据库已正常。

退出hive命令行, 使用exit;

## 设置直连数据库模式

直连数据库模式也叫本地模式, 它和远程模式安装配置方式大致相同, 本质上是将Hive默认的元数据存储介质由自带的Derby数据库替换为MySQL数据库。无论在什么目录下以任何方式启动Hive, 只要连接的是同一台Hive服务, 所有节点访问的元数据信息均是一致的, 从而实现元数据的共享。以直连数据库模式为例, 讲解Hive的安装过程。直连数据库模式的Hive安装主要包括安装MySQL服务和安装配置Hive两个步骤。

以下在master主机上执行:

### 安装Mysql和配置mysql库

删除已有mysql安装新mysql

```

1 rm -rf /usr/lib/mysql
2 rm -rf /usr/include/mysql
3 rm -rf /etc/my.cnf
4 rm -rf /var/lib/mysql
5 rm -rf /usr/share/mysql
6 cd /root
7 yum remove -y mariadb-libs
8 yum install -y net-tools
9 yum install -y perl
10 yum remove mysql mysql-server
11 yum install -y mariadb-libs
12 rpm -ivh mysql-community-* --force --nodeps
13 cp /root/mysql-connector-java-5.1.32-bin.jar /usr/local/hive/lib/

```

## 修改mysql配置

```
1 vi /etc/my.cnf
```

添加如下内容:

```

1 character_set_server=utf8
2 [client]
3 default-character-set=utf8
4 [mysql]
5 default-character-set=utf8

```

my.cnf文件结果如:

```

# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.7/en/server-configuration-defaults.html

[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

character_set_server=utf8
[client]
default-character-set=utf8
[mysql]
default-character-set=utf8
~
~
~

```

## 启mysql服务

```
1 | systemctl start mysqld
2 | systemctl enable mysqld
3 | service mysqld restart
```

## 查看临时密码

```
1 | more /var/log/mysqld.log |grep password |grep generated
```

结果如下，其中:"K6UdV+XTmoAB"，为本机临时密码，每台机器密码都不同

```
1 | 2023-11-08T03:46:49.898572Z 1 [Note] A temporary password is generated for
    root@localhost: K6UdV+XTmoAB
2 | [root@master ~]
```

## 使用临时密码登陆:

```
1 | mysql -uroot -p'K6UdV+XTmoAB'
```

-p后的临时密码需要替换成本机查询出来的密码。

## 修改mysql的root密码

进入mysql命令后，修改mysql的root密码为:123456

```
1 | alter user 'root'@'localhost' identified by '@Root_123456';
2 | set global validate_password_policy=0;
3 | set global validate_password_length=6;
4 | set password for 'root'@'localhost'=password('123456');
5 | GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT
    OPTION;
6 | FLUSH PRIVILEGES;
7 | create database hive;
8 | create database hive_remote;
9 | exit;
```

运行结果如:

```
1 | mysql> alter user 'root'@'localhost' identified by '@Root_123456';
2 | Query OK, 0 rows affected (0.00 sec)
3 |
4 | mysql> set global validate_password_policy=0;
5 | Query OK, 0 rows affected (0.00 sec)
6 |
7 | mysql> set global validate_password_length=6;
8 | Query OK, 0 rows affected (0.00 sec)
9 |
10 | mysql> set password for 'root'@'localhost'=password('123456');
11 | Query OK, 0 rows affected, 1 warning (0.00 sec)
12 |
13 | mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH
    GRANT OPTION;
```

```
14 Query OK, 0 rows affected, 1 warning (0.00 sec)
15
16 mysql> FLUSH PRIVILEGES;
17 Query OK, 0 rows affected (0.00 sec)
18
19 mysql> exit;
```

## 配置Hive直连数据库

仍在master配置，所以安装hive过程，/etc/profile环境变量配置不变。

### 需要配置hive-site.xml

```
1 cd /usr/local/hive/conf/
2 vi ./hive-site.xml
```

hive-site.xml内容设置如下：

```
1 <configuration>
2 <property>
3 <name>javax.jdo.option.ConnectionURL</name>
4 <value>jdbc:mysql://master:3306/hive?
  createDatabaseIfNotExist=true&amp;useSSL=false&amp;useUnicode=true&amp;chara
  cterEncoding=UTF-8</value>
5 </property>
6 <property>
7 <name>javax.jdo.option.ConnectionDriverName</name>
8 <value>com.mysql.jdbc.Driver</value>
9 <description>Driver class name for a JDBC metastore</description>
10 </property>
11 <property>
12 <name>javax.jdo.option.ConnectionUserName</name>
13 <value>root</value>
14 <description>Username to use against metastore database</description>
15 </property>
16 <property>
17 <name>javax.jdo.option.ConnectionPassword</name>
18 <value>123456</value>
19 <description>password to use against metastore database</description>
20 </property>
21 <property>
22 <name>hive.metastore.event.db.notification.api.auth</name>
23 <value>false</value>
24 </property>
25 <property>
26 <name>hive.metastore.schema.verification</name>
27 <value>false</value>
28 </property>
29 </configuration>
```

(! 请注意，第4行和5行之前是没有换行的，同学们需要在记事本同合并一行)

## 初始化元数据库

```
1 | cd /usr/local/hive/bin
2 | ./schematool -dbType mysql -initSchema
```

运行后会出现如下completed的内容，说明元数据库初始化完成

```
1 | Initialization script completed
2 | schemaTool completed
3 | [root@master bin]#
```

## 运行hive

(前提是hadoop集群已启动，使用jps查看，若没有，使用start-all.sh启动hadoop集群)

```
1 | hive
```

运行结果如:

```
1 | [root@master bin]# hive
2 | SLF4J: Class path contains multiple SLF4J bindings.
3 | SLF4J: Found binding in [jar:file:/usr/local/hadoop-
4 | 3.1.4/share/hadoop/common/lib/slf4j-log4j12-
5 | 1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
6 | SLF4J: Found binding in [jar:file:/usr/local/hadoop-
7 | 3.1.4/share/hadoop/common/slf4j-log4j12-
8 | 1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
9 | SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
10 | explanation.
11 | SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
12 | which: no hbase in (/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
13 | 3.1.4/bin:/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
14 | 3.1.4/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.8
15 | .0_281-amd64/bin:/root/bin:/usr/java/jdk1.8.0_281-
16 | amd64/bin:/bin:/usr/local/hive/bin:/usr/java/jdk1.8.0_281-
17 | amd64/bin:/usr/local/hive/bin)
18 | SLF4J: Class path contains multiple SLF4J bindings.
19 | SLF4J: Found binding in [jar:file:/usr/local/hadoop-
20 | 3.1.4/share/hadoop/common/lib/slf4j-log4j12-
21 | 1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
22 | SLF4J: Found binding in [jar:file:/usr/local/hadoop-
23 | 3.1.4/share/hadoop/common/slf4j-log4j12-
24 | 1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
25 | SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
26 | explanation.
27 | SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
28 | Hive Session ID = 12077e17-f378-4958-8790-dba9205eec27
29 |
30 | Logging initialized using configuration in
31 | jar:file:/usr/local/hive/lib/hive-common-3.1.2.jar!/hive-log4j2.properties
32 | Async: true
```

```
16 | Hive-on-MR is deprecated in Hive 2 and may not be available in the future
    | versions. Consider using a different execution engine (i.e. spark, tez) or
    | using Hive 1.X releases.
17 | Hive Session ID = 7bcad5dd-93b8-473b-a126-ed08898be1fa
18 | hive>
19 | hive> show databases;
20 | OK
21 | default
22 | Time taken: 0.851 seconds, Fetched: 1 row(s)
23 | hive> exit;
```

至此, 完成了Hive直连数据库的模式。

## 设置远程模式

### 修改master上的配置文件hive-site.xml

```
1 | cd /usr/local/hive/conf/
2 | cp ./hive-site.xml ./hive-site.xml.direct
3 | vi ./hive-site.xml
```

hive-site.xml修改内容如下:

其中有修改行8, 将hive库改为hive\_remote:

jdbc:mysql://master:3306/hive\_remote?

createDatabaseIfNotExist=true&useSSL=false&useUnicode=true&characterEncoding=UTF-8

```
1 | <configuration>
2 | <property>
3 | <name>hive.metastore.warehouse.dir</name>
4 | <value>/user/hive_remote/warehouse</value>
5 | </property>
6 | <property>
7 | <name>javax.jdo.option.ConnectionURL</name>
8 | <value>jdbc:mysql://master:3306/hive_remote?
   | createDatabaseIfNotExist=true&useSSL=false&useUnicode=true&chara
   | cterEncoding=UTF-8</value>
9 | </property>
10 | <property>
11 | <name>javax.jdo.option.ConnectionDriverName</name>
12 | <value>com.mysql.jdbc.Driver</value>
13 | <description>Driver class name for a JDBC metastore</description>
14 | </property>
15 | <property>
16 | <name>javax.jdo.option.ConnectionUserName</name>
17 | <value>root</value>
18 | <description>Username to use against metastore database</description>
19 | </property>
20 | <property>
21 | <name>javax.jdo.option.ConnectionPassword</name>
22 | <value>123456</value>
23 | <description>password to use against metastore database</description>
24 | </property>
25 | </property>
```

```
26 <name>datanucleus.schema.autoCreateAll</name>
27 <value>true</value>
28 </property>
29 </property>
30 <name>hive.metastore.schema.verification</name>
31 <value>false</value>
32 </property>
33 </configuration>
```

## 再次初始化元数据库

```
1 cd /usr/local/hive/bin
2 ./schematool -dbtype mysql -initSchema
```

运行后会出现如下completed的内容，说明元数据库初始化完成

```
1 Initialization script completed
2 schemaTool completed
3 [root@master bin]#
```

## 从master分发hive到从机slave1,slave2

```
1 scp -r /usr/local/hive slave1:/usr/local/
2 scp -r /usr/local/hive slave2:/usr/local/
3 scp /etc/profile slave1:/etc/
4 scp /etc/profile slave2:/etc/
```

## 在从机slave1和 slave2上执行

```
1 source /etc/profile
```

## 在从机slave1和slave2上修改hive-site.xml文件

```
1 cd /usr/local/hive/conf/
2 vi ./hive-site.xml
```

hive-site.xml内容如下：

```
1 <configuration>
2 <property>
3 <name>hive.metastore.warehouse.dir</name>
4 <value>/user/hive_remote/warehouse</value>
5 </property>
6 <property>
7 <name>hive.metastore.uris</name>
8 <value>thrift://master:9083</value>
9 </property>
10 </configuration>
```

## 在master上启动hive服务

```
1 | hive --service metastore &
```

(&表示后台运行, 不加&将在前台运行, 这时不能进行其它操作, Ctrl+C 将退出hive服务)

创建shell文件执行

```
1 | vi /root/start-hive.sh
2 | chmod 777 /root/start-hive.sh
```

start-hive.sh的内容如:

```
1 | hive --service metastore
```

后面启动master的hive服务就可以使用

```
1 | cd /root
2 | ./start-hive.sh
```

## 在slave1或slave2上运行hive

```
1 | hive
```

结果如:

```
1 | [root@slave2 local]# hive
2 | which: no hbase in (/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
3 | 3.1.4/bin:/usr/local/hadoop-3.1.4/sbin:/usr/local/hadoop-
4 | 3.1.4/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/java/jdk1.8
5 | .0_281-amd64/bin:/root/bin:/usr/java/jdk1.8.0_281-
6 | amd64/bin:/usr/java/jdk1.8.0_281-amd64/bin:/usr/local/hive/bin)
7 | Hive session ID = 8f86b1cb-026f-4105-9ced-086bb53bd27a
8 |
9 | Logging initialized using configuration in
10 | jar:file:/usr/local/hive/lib/hive-common-3.1.2.jar!/hive-log4j2.properties
11 | Async: true
12 | Hive-on-MR is deprecated in Hive 2 and may not be available in the future
13 | versions. Consider using a different execution engine (i.e. spark, tez) or
14 | using Hive 1.X releases.Hive session ID = 5c2f0684-fd26-4ff4-b78b-
15 | 5b7e8c1e4641
16 |
17 | hive> show databases;
18 | OK
19 | default
20 | Time taken: 0.618 seconds, Fetched: 1 row(s)
21 | hive>
```

## 在slave1或slave2创建hive\_remote数据库

```
1 | create database hive_remote;
```

运行结果:

```
1 | hive> create database hive_remote;
2 | OK
3 | Time taken: 1.196 seconds
4 | hive> show databases;
5 | OK
6 | default
7 | hive_remote
8 | Time taken: 0.043 seconds, Fetched: 2 row(s)
9 | hive>
```

## 在master 的mysql库观察

有新增hive\_remote数据库

```
1 | mysql -uroot -p123456
```

如

```
1 | [root@master ~]# mysql -uroot -p123456
2 | mysql: [warning] Using a password on the command line interface can be
   | insecure.
3 | Welcome to the MySQL monitor.  Commands end with ; or \g.
4 | Your MySQL connection id is 275
5 | Server version: 5.7.18 MySQL Community Server (GPL)
6 |
7 | Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.
8 |
9 | Oracle is a registered trademark of Oracle Corporation and/or its
10 | affiliates. Other names may be trademarks of their respective
11 | owners.
12 |
13 | Type 'help;' or '\h' for help. Type '\c' to clear the current input
   | statement.
14 |
15 | mysql>
```

查看数据库

```

1  mysql> show databases;
2  +-----+
3  | Database           |
4  +-----+
5  | information_schema |
6  | hive                |
7  | hive_remote        |
8  | mysql              |
9  | performance_schema |
10 | sys                 |
11 +-----+
12 6 rows in set (0.00 sec)

```

### 修改数据库的字符集

为后续创建hive表准备

```

1  ALTER DATABASE hive CHARACTER SET latin1;
2  ALTER DATABASE hive_remote CHARACTER SET latin1;

```

## 任务6.3实现Hive表的创建与修改

尽管HiveQL的语法规则与SQL语法非常相似，但是，Hive在模式设计上与传统关系型数据库非常不同。Hive是反模式的，且在反模式的应用中，与传统数据库也是有区别的。

本小节的任务如下。

1. 了解Hive的基本语法。
2. 掌握Hive表的创建与修改。

### Hive数据定义语言DDL的基本语法

Hive数据定义语言（DDL）可以实现Hive数据存储所依赖的数据库及表结构的定义、表描述查看和表结构更改的操作。与传统数据库（如MySQL）用法类似，Hive需要先创建数据库，再创建表，表中有表结构，数据按表结构进行存储。与传统数据库相比，Hive对表格式的定义更加宽松、随性。

数据库的基本操作

```

1  使用CREATE命令可以创建数据库，创建数据库语法格式如下。
2  CREATE (DATABASE | SCHEMA) [IF NOT EXISTS] database_name
3     [COMMENT database_comment]
4     [LOCATION hdfs_path]
5     [ WITHDBPROPERTIES (property_name=property_value,...)];
6
7  使用DROP命令可以删除数据库，删除数据库语法格式如下。
8  DROP(DATABASE | SCHEMA) [IF EXISTS] database_name [RESTRICTI | CASCADE];
9
10 使用ALTER命令可以更改数据库当前目录，更改数据库语法格式如下。
11 ALTER (DATABASE | SCHEMA) database_name SET LOCATION hdfs_path;
12 --(Note: Hive2.2.1,2.4.and later)
13
14使用数据库语法格式如下。
15 USE database_name;

```

```
16 USE DEPAULT;  
17
```

## 表的基本操作

```
1 其中Hive中创建表的基本语法格式如下。  
2 CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name  
3     [(col_name data_type [COMMENT col_comment], ...)]  
4     [COMMENT table_comment]  
5     [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]  
6     [CLUSTERED BY (col_name, col_name, ...)]  
7     [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]  
8     [ROW FORMAT row_format]  
9     [STORED AS file_format]  
10    [LOCATION hdfs_path]  
11  
12 Hive中复制表的基本语法格式如下。  
13 CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name  
14 LIKE existing_table_or_view_name [LOCATION hdfs_path]  
15  
16 Hive中删除表的基本语法格式如下。  
17 DROP TABLE [IF EXISTS] table_name [PURGE]  
18 -- (Note: PURGE available in Hive 0.14.0 and later)  
19  
20 通过RENAME语句可以将表的名称更改为其他名称，语法格式如下。  
21 ALTER TABLE table_name RENAME TO new_table_name;  
22  
23 通过ALTER TABLE语句可以添加或删除表约束，语法格式如下。  
24 ALTER TABLE table_name ADD CONSTRAINT constraint_name PRIMARY KEY (column,  
25 ... ) DISABLE NOVALIDATE;  
26 ALTER TABLE table_name ADD CONSTRAINT constraint_name FOREIGN KEY (column,  
27 ... ) REFERENCES table_name(column, ...) DISABLE NOVALIDATE RELY;  
28 ALTER TABLE table_name CHANGE COLUMN column_name column_name data_type  
29 CONSTRAINT constraint_name NOT NULL ENABLE;  
30 ALTER TABLE table_name DROP CONSTRAINT constraint_name;  
31  
32 通过ALTER TABLE语句也可以添加或删除表的分区，语法格式如下。  
33 ALTER TABLE table_name ADD [IF NOT EXISTS] PARTITION partition_spec  
34 [LOCATION 'location'][, PARTITION partition_spec [LOCATION 'location'],  
35 ...];  
36 partition_spec:  
37 (partition_column = partition_col_value, partition_column =  
38 partition_col_value, ...)  
39 ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec,  
40 partition_spec, ...]  
41 [IGNORE PROTECTION] [PURGE];
```

## Hive表的基本操作

准备测试数据，如果前面已完成数据准备，这里上传指令可以忽略，在master或slave上执行都可以：

```
1 | cd /root/hadoop
2 | tar -zxvf ./hadoop_data.tar.gz
3 | hdfs dfs -mkdir /user/myname/stu
4 | hdfs dfs -put ./hadoop_data/student_info.txt /user/myname/stu
```

修改mysql字符集，在master主机上修改；不修改的话在创建hive表会报错：Column length too big for column 'TYPE\_NAME'

```
1 | ALTER DATABASE hive CHARACTER SET latin1;
2 | ALTER DATABASE hive_remote CHARACTER SET latin1;
```

运行结果如：

```
1 | mysql> ALTER DATABASE hive CHARACTER SET latin1;
2 | Query OK, 1 row affected (0.00 sec)
3 |
4 | mysql> ALTER DATABASE hive_remote CHARACTER SET latin1;
5 | Query OK, 1 row affected (0.00 sec)
6 |
7 | mysql> exit;
```

表的基本操作：内部表，外部表，分区表；在slave1上的hive命令行执行：

```
1 | -- 创建内部表
2 | create database student;
3 | use student;
4 | create table user_info(id int,name string);
5 | -- 创建外部表
6 | create external table student_info(
7 | stu_no string,
8 | stu_name string,
9 | stu_sex string,
10 | telephone string,
11 | stu_class string)
12 | row format delimited fields terminated by ',' location '/user/myname/stu';
13 | show tables;
14 | select * from user_info;
15 | select * from student_info;
16 | desc student_info;
17 | -- 创建分区表
18 | create table teacher_info(t_no string,t_name string,t_sex string,t_age
19 | string)
20 | partitioned by (depart string)
21 | row format delimited fields terminated by ',' stored as textfile;
22 | -- 创建表score
23 | create table score(
24 | stu_no string,
25 | cla_no string ,
26 | grade float)
```

```

26 | partitioned by (class_name string);
27 | -- 修改表
28 | alter table score rename to stu_score;
29 | -- 修改表, 添加字段
30 | alter table stu_score add columns (credit int, gpa float);
31 | -- 修改表, 修改字段
32 | alter table stu_score change column credit Credits float;
33 | -- 修改表, 新增分区
34 | alter table stu_score add partition(class_name ='20231109');
35 | -- 修改表, 删除分区
36 | alter table stu_score drop if exists partition(class = '20231109');
37 | -- 删除表
38 | drop table stu_score;

```

运行结果如:

```

1 | hive> create table user_info (id int ,name string);
2 | OK
3 | Time taken: 0.642 seconds
4 | hive> -- 创建外部表
5 | hive> create external table student_info(
6 |     > stu_no string,
7 |     > stu_name string,
8 |     > stu_sex string,
9 |     > telephone string,
10 |     > stu_class string)
11 |     > row format delimited fields terminated by ',' location
12 |     '/user/myname/stu';
13 | OK
14 | Time taken: 0.13 seconds
15 | hive>
16 | hive> show tables;
17 | OK
18 | student_info
19 | user_info
20 | Time taken: 0.04 seconds, Fetched: 2 row(s)
21 | hive> select * from user_info;
22 | OK
23 | Time taken: 1.945 seconds
24 | hive> select * from student_info;
25 | OK
26 | stu_no  stu_name      stu_sex telephone      stu_class
27 | 2021513501  张晓娟  女      153*****506  2021级信管1班
28 | 2021513505  李小鹏  男      156*****501  2021级信管2班
29 | 2021513503  张丽    女      186*****556  2021级信管1班
30 | 2021513504  刘向东  男      133*****314  2021级信管2班
31 | 2021513505  李明博  男      137*****049  2021级信管1班
32 | 2021513506  张子强  男      135*****379  2021级信管2班
33 | 2021513507  王文海  男      130*****410  2021级信管1班
34 | 2021513500  刘慧娟  女      130*****140  2021级信管2班
35 | 2021513509  陈敏    女      133*****550  2021级信管1班
36 | 2021513510  王云    男      100*****475  2021级信管2班
37 | 2021513511  沈璐    女      156*****675  2021级信管1班
38 | 2021513515  赵鹏飞  男      133*****514  2021级信管3班
39 | 2021513513  谢朝阳  男      153*****557  2021级信管4班

```

```

39 2021513514 刘子歌 女 135*****533 2021级信管3班
40 2021513515 陈照升 男 130*****500 2021级信管1班
41 2021513516 秦小路 女 153*****505 2021级信管3班
42 2021513517 张大宝 男 133*****541 2021级信管4班
43 2021513510 刘佳 女 133*****453 2021级信管3班
44 2021513519 李伟 男 133*****571 2021级信管1班
45 2021513550 杨荣 女 151*****006 2021级信管3班
46 2021513551 李大双 男 131*****503 2021级信管1班
47 2021513555 赵晓雨 女 109*****050 2021级信管5班
48 2021513553 王大红 女 150*****740 2021级信管4班
49 2021513554 张雨滴 女 176*****059 2021级信管5班
50 2021513555 李斯 女 150*****559 2021级信管4班
51 2021513556 秦小宝 男 153*****056 2021级信管5班
52 2021513557 魏中祥 男 177*****360 2021级信管4班
53 2021513550 于媛媛 女 157*****156 2021级信管5班
54 2021513559 张耗 男 130*****373 2021级信管4班
55 2021513530 张春晓 女 177*****903 2021级信管5班
56 2021513531 章小玉 女 153*****459 2021级信管4班
57 2021513535 林飞飞 女 177*****774 2021级信管5班
58 2021513533 朱云云 女 173*****779 2021级信管4班
59 2021513534 夏晓晓 女 135*****550 2021级信管5班
60 Time taken: 0.177 seconds, Fetched: 35 row(s)
61 hive>
62 hive> desc student_info;
63 OK
64 stu_no string
65 stu_name string
66 stu_sex string
67 telephone string
68 stu_class string
69 Time taken: 0.057 seconds, Fetched: 5 row(s)
70 hive>
71 hive> -- 创建分区表
72 hive> create table teacher_info(t_no string,t_name string,t_sex
string,t_age string)
73 > partitioned by (deppart string)
74 > row format delimited fields terminated by ',' stored as textfile;
75 OK
76 Time taken: 0.113 seconds
77 hive>
78 hive> -- 创建表score
79 hive> create table score(
80 > stu_no string,
81 > cla_no string ,
82 > grade float)
83 > partitioned by (class_name string);
84 OK
85 Time taken: 0.11 seconds
86 hive>
87 hive> -- 修改表
88 hive> alter table score rename to stu_score;
89 OK
90 Time taken: 0.253 seconds
91 hive>
92 hive> -- 修改表, 添加字段

```

```

93 | hive> alter table stu_score add columns (credit int, gpa float);
94 | OK
95 | Time taken: 0.141 seconds
96 | hive>
97 | hive> -- 修改表, 修改字段
98 | hive> alter table stu_score change column credit Credits float;
99 | OK
100 | Time taken: 0.126 seconds
101 | hive>
102 | hive> -- 修改表, 新增分区
103 | hive> alter table stu_score add partition(class_name = '20231109');
104 | OK
105 | Time taken: 0.274 seconds
106 | hive> -- 修改表, 删除分区
107 | hive> alter table stu_score drop if exists partition(class = '20231109');
108 | FAILED: NullPointerException null
109 | hive> -- 删除表
110 | hive> drop table stu_score;
111 | OK
112 | Time taken: 0.357 seconds
113 | hive>

```

## 任务6.4实现Hive表中数据的增删查改

### Hive数据操作语言DML的基本语法

```

1 | -- 装载数据。
2 | LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
3 | [PARTITION
4 | (partcol1=val1, partcol2=val2 ...)]
5 | LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
6 | [PARTITION
7 | (partcol1=val1, partcol2=val2 ...)] [INPUTFORMAT 'inputformat' SERDE
8 | 'serde']
9 |
10 | -- 查询数据格式如下。
11 | SELECT [ALL | DISTINCT] select_expr, select_expr, ...
12 | FROM table_reference
13 | [WHERE where_condition]
14 | [GROUP BY col_list]
15 | [ORDER BY col_list]
16 | [CLUSTER BY col_list]
17 | | [DISTRIBUTE BY col_list] [SORT BY col_list]
18 | ]
19 | [LIMIT [offset,] rows]
20 |
21 | -- 插入数据格式如下。
22 | INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2
23 | ...) [IF NOT EXISTS]] select_statement1 FROM from_statement;
24 | INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)]
25 | select_statement1 FROM from_statement;

```

```
22 -- 删除数据格式如下。
23 DELETE FROM tablename [WHERE expression];
24
```

## Hive表的基本数据操作

使用LOAD命令将hdfs上的3个文件的数据分别装载至Hive中的3个表中

参考数据准备时已将数据文件导入到hdfs:

```
1 hdfs dfs -put ./hadoop_data/student_info.txt /user/myname/stu
2 hdfs dfs -put ./hadoop_data/students.txt /user/myname/stu
3 hdfs dfs -put ./hadoop_data/course.txt /user/myname/stu
4 hdfs dfs -put ./hadoop_data/sc.txt /user/myname/stu
```

以下在 slave1或slave2中执行load命令

```
1 -- 创建数据表
2 create table student(Sno int,Sname string,Sex string,Sage int,Sdept string)
3 row format delimited fields terminated by ',' stored as textfile;
4 create table course(Cno int,Cname string)
5 row format delimited fields terminated by ',' stored as textfile;
6 create table sc(Sno int,Cno int,Grade int)
7 row format delimited fields terminated by ',' stored as textfile;
8 -- 向数据表中装载文件
9 load data inpath '/user/myname/stu/students.txt' overwrite into table
  student;
10 load data inpath '/user/myname/stu/course.txt' overwrite into table course;
11 load data inpath '/user/myname/stu/sc.txt' overwrite into table sc;
12 -- 查询数据
13 select Sno,Sname from student;
14 select Sno,Sname from student where Sex='男';
15 select distinct Sname from student inner join sc on student.Sno=sc.Sno;
16 -- 查询数据, 查询学生的课程成绩情况
17 select student.Sname,Course.Cname,Sc.Grade from student join sc
18 on student.Sno=sc.Sno join course on sc.cno=course.cno;
19 -- 查询数据, 查询选修2号课程且成绩为90分以上的学生
20 select student.Sname,sc.Grade from student join sc
21 on student.Sno=sc.Sno and sc.Grade>90;
22 -- 查询数据, 查询各个课程及相应的选课人数
23 select Cno,count(1) from sc group by Cno;
24 -- 查询数据, 查询选修了3门以上的课程的学生学号
25 select Sno from sc group by Sno having count(Cno)>3;
26 -- 查询数据, 查询学生信息并根据学号进行升序排序
27 select Sno from student order by Sno;
28 -- 查询数据, 按性别分区, 在分区内按年龄升序排序
29 set mapred.reduce.tasks=2;
30 insert overwrite directory '/user/myname/stu_out'
31 select * from student
32 distribute by Sex sort by Sage;
33 -- 插入数据
34 -- 插入数据,在student表中插入一条新记录
35 insert into table student values(2018213223,'王小哲','男',18,'IS');
36 -- 删除数据
```

```
37 | delete from student where Sno='2018213223';
```

## 运行结果

```
1 | hive> -- 创建数据表
2 | hive> create table student(Sno int,Sname string,Sex string,Sage int,Sdept
   | string)
3 |     > row format delimited fields terminated by ',' stored as textfile;
4 | OK
5 | Time taken: 0.137 seconds
6 | hive> create table course(Cno int,Cname string)
7 |     > row format delimited fields terminated by ',' stored as textfile;
8 | OK
9 | Time taken: 0.112 seconds
10 | hive> create table sc(Sno int,Cno int,Grade int)
11 |     > row format delimited fields terminated by ',' stored as textfile;
12 | OK
13 | Time taken: 0.061 seconds
14 | hive>
15 | hive> -- 向数据表中装载文件
16 | hive> load data inpath '/user/myname/stu/students.txt' overwrite into table
   | student;
17 | Loading data to table student.student
18 | OK
19 | Time taken: 0.376 seconds
20 | hive>
21 | hive> load data inpath '/user/myname/stu/course.txt' overwrite into table
   | course;
22 | Loading data to table student.course
23 | OK
24 | Time taken: 0.252 seconds
25 | hive> load data inpath '/user/myname/stu/sc.txt' overwrite into table sc;
26 | Loading data to table student.sc
27 | OK
28 | Time taken: 0.276 seconds
29 | hive>
30 | hive> -- 查询数据
31 | hive> select Sno,Sname from student;
32 | OK
33 | 2018213201      李小勇
34 | 2018213202      刘良
35 | 2018213203      王芝芝
36 | 2018213204      张大立
37 | 2018213205      刘云山
38 | 2018213206      孙庆刚
39 | 2018213207      易思玲
40 | 2018213208      李娜
41 | 2018213209      梦媛媛
42 | 2018213210      孔江涛
43 | 2018213211      包小波
44 | 2018213212      孙晓花
45 | 2018213213      冯伟伟
46 | 2018213214      王小容
47 | 2018213216      王君丹
48 | 2018213215      钱国峰
```

```
49 2018213217 王风娟
50 2018213218 王一一
51 2018213219 邢小雨
52 2018213220 赵钱钱
53 2018213221 周小二
54 2018213222 郑明山
55 Time taken: 0.134 seconds, Fetched: 22 row(s)
56 hive> select Sno,Sname from student where Sex='男';
57 OK
58 2018213201 李小勇
59 2018213204 张大立
60 2018213205 刘云山
61 2018213206 孙庆刚
62 2018213210 孔江涛
63 2018213211 包小波
64 2018213213 冯伟伟
65 2018213216 王君丹
66 2018213215 钱国峰
67 2018213220 赵钱钱
68 2018213221 周小二
69 2018213222 郑明山
70 Time taken: 0.139 seconds, Fetched: 12 row(s)
71 hive> select distinct Sname from student inner join sc on
student.Sno=sc.Sno;
72 Query ID = root_20231109140019_1e25a730-12e3-439b-9c74-0fdadb7b6244
73 Total jobs = 1
74 2023-11-09 14:00:30 Dump the side-table for tag: 0 with group count: 22
into file: file:/tmp/root/6c7f3d1e-927d-4f70-88e4-09aeca3d0443/hive_2023-
11-09_14-00-19_828_5670865665580644800-1/-local-10005/HashTable-Stage-
2/MapJoin-mapfile00--.hashtable
75 2023-11-09 14:00:30 Uploaded 1 File to: file:/tmp/root/6c7f3d1e-927d-
4f70-88e4-09aeca3d0443/hive_2023-11-09_14-00-19_828_5670865665580644800-1/-
local-10005/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (983 bytes)
76 2023-11-09 14:00:30 End of local task; Time Taken: 1.637 sec.
77 Execution completed successfully
78 MapredLocal task succeeded
79 Launching Job 1 out of 1
80 Number of reduce tasks not specified. Estimated from input data size: 1
81 In order to change the average load for a reducer (in bytes):
82 set hive.exec.reducers.bytes.per.reducer=<number>
83 In order to limit the maximum number of reducers:
84 set hive.exec.reducers.max=<number>
85 In order to set a constant number of reducers:
86 set mapreduce.job.reduces=<number>
87 Starting Job = job_1699491849504_0001, Tracking URL =
http://master:8088/proxy/application_1699491849504_0001/
88 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0001
89 Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
90 2023-11-09 14:00:43,702 Stage-2 map = 0%, reduce = 0%
91 2023-11-09 14:00:54,068 Stage-2 map = 100%, reduce = 0%, Cumulative CPU
2.85 sec
92 2023-11-09 14:01:00,215 Stage-2 map = 100%, reduce = 100%, Cumulative CPU
4.83 sec
```

```
93 MapReduce Total cumulative CPU time: 4 seconds 830 msec
94 Ended Job = job_1699491849504_0001
95 MapReduce Jobs Launched:
96 Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.83 sec HDFS Read:
18207 HDFS write: 757 SUCCESS
97 Total MapReduce CPU Time Spent: 4 seconds 830 msec
98 OK
99 冯伟伟
100 刘云山
101 刘良
102 包小波
103 周小二
104 孔江涛
105 孙庆刚
106 孙晓花
107 张大立
108 易思玲
109 李娜
110 李小勇
111 梦媛媛
112 王一一
113 王君丹
114 王小容
115 王芝芝
116 王凤娟
117 赵钱钱
118 邢小雨
119 郑明山
120 钱国峰
121 Time taken: 42.522 seconds, Fetched: 22 row(s)
122 hive>
123 hive> -- 查询数据, 查询学生的课程成绩情况
124 hive> select student.Sname, Course.Cname, Sc.Grade from student join sc
125 > on student.Sno=sc.Sno join course on sc.cno=course.cno;
126 No Stats for student@student, Columns: sno, sname
127 No Stats for student@sc, Columns: sno, cno, grade
128 No Stats for student@course, Columns: cno, cname
129 Query ID = root_20231109140520_f5da6c0d-77b7-4fbe-ad10-06cdb071c284
130 Total jobs = 1
131 Execution completed successfully
132 MapredLocal task succeeded
133 Launching Job 1 out of 1
134 Number of reduce tasks is set to 0 since there's no reduce operator
135 Starting Job = job_1699491849504_0002, Tracking URL =
http://master:8088/proxy/application_1699491849504_0002/
136 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0002
137 Hadoop job information for Stage-5: number of mappers: 1; number of
reducers: 0
138 2023-11-09 14:05:39,581 Stage-5 map = 0%, reduce = 0%
139 2023-11-09 14:05:46,821 Stage-5 map = 100%, reduce = 0%, Cumulative CPU
2.2 sec
140 MapReduce Total cumulative CPU time: 2 seconds 200 msec
141 Ended Job = job_1699491849504_0002
142 MapReduce Jobs Launched:
```

```

143 Stage-Stage-5: Map: 1 Cumulative CPU: 2.2 sec HDFS Read: 12571 HDFS
Write: 6514 SUCCESS
144 Total MapReduce CPU Time Spent: 2 seconds 200 msec
145 OK
146 李小勇 Python语言程序设计 81
147 李小勇 数据库系统原理 85
148 李小勇 信息系统分析与设计 88
149 李小勇 大数据技术及应用 70
150 刘良 数据库系统原理 90
151 刘良 信息系统分析与设计 80
152 刘良 大数据技术及应用 71
153 刘良 Hadoop开发与应用 60
154 王芝芝 Python语言程序设计 82
155 王芝芝 信息系统分析与设计 90
156 王芝芝 Hadoop开发与应用 100
157 张大立 Python语言程序设计 80
158 张大立 数据库系统原理 92
159 张大立 大数据技术及应用 91
160 张大立 Hadoop开发与应用 70
161 刘云山 Python语言程序设计 70
162 刘云山 数据库系统原理 92
163 刘云山 信息系统分析与设计 99
164 刘云山 web应用程序设计 87
165 孙庆刚 Python语言程序设计 72
166 孙庆刚 数据库系统原理 62
167 孙庆刚 信息系统分析与设计 100
168 孙庆刚 大数据技术及应用 59
169 孙庆刚 Hadoop开发与应用 60
170 孙庆刚 web应用程序设计 98
171 易思玲 信息系统分析与设计 68
172 易思玲 大数据技术及应用 91
173 易思玲 Hadoop开发与应用 94
174 易思玲 web应用程序设计 78
175 李娜 Python语言程序设计 98
176 李娜 信息系统分析与设计 89
177 李娜 web应用程序设计 91
178 梦媛媛 数据库系统原理 81
179 梦媛媛 大数据技术及应用 89
180 梦媛媛 web应用程序设计 100
181 孔江涛 数据库系统原理 98
182 孔江涛 Hadoop开发与应用 90
183 孔江涛 web应用程序设计 80
184 包小波 Python语言程序设计 81
185 包小波 数据库系统原理 91
186 包小波 信息系统分析与设计 81
187 包小波 大数据技术及应用 86
188 孙晓花 Python语言程序设计 81
189 孙晓花 信息系统分析与设计 78
190 孙晓花 大数据技术及应用 85
191 孙晓花 web应用程序设计 98
192 冯伟伟 Python语言程序设计 98
193 冯伟伟 数据库系统原理 58
194 冯伟伟 大数据技术及应用 88
195 冯伟伟 Hadoop开发与应用 93
196 王小容 Python语言程序设计 91

```

197	王小容	数据库系统原理	100
198	王小容	大数据技术及应用	98
199	钱国峰	Python语言程序设计	91
200	钱国峰	信息系统分析与设计	59
201	钱国峰	大数据技术及应用	100
202	钱国峰	web应用程序设计	95
203	王君丹	Python语言程序设计	92
204	王君丹	数据库系统原理	99
205	王君丹	大数据技术及应用	82
206	王风娟	大数据技术及应用	82
207	王风娟	Hadoop开发与应用	100
208	王风娟	web应用程序设计	58
209	王一一	Python语言程序设计	95
210	王一一	数据库系统原理	100
211	王一一	信息系统分析与设计	67
212	王一一	大数据技术及应用	78
213	邢小雨	Python语言程序设计	77
214	邢小雨	数据库系统原理	90
215	邢小雨	信息系统分析与设计	91
216	邢小雨	大数据技术及应用	67
217	邢小雨	Hadoop开发与应用	87
218	赵钱钱	Python语言程序设计	66
219	赵钱钱	数据库系统原理	99
220	赵钱钱	Hadoop开发与应用	93
221	周小二	数据库系统原理	93
222	周小二	Hadoop开发与应用	91
223	周小二	web应用程序设计	99
224	郑明山	信息系统分析与设计	69
225	郑明山	大数据技术及应用	93
226	郑明山	Hadoop开发与应用	82
227	郑明山	web应用程序设计	100
228	Time taken: 28.546 seconds, Fetched: 82 row(s)		
229	hive> -- 查询数据, 查询选修2号课程且成绩为90分以上的学生		
230	hive> select student.Sname,sc.Grade from student join sc		
231	> on student.Sno=sc.Sno and sc.Grade>90;		
232	Query ID = root_20231109140552_8e9aa7db-e4c9-477e-8357-75851f1d8546		
233	Total jobs = 1		
234	Execution completed successfully		
235	MapredLocal task succeeded		
236	Launching Job 1 out of 1		
237	Number of reduce tasks is set to 0 since there's no reduce operator		
238	Starting Job = job_1699491849504_0003, Tracking URL =		
	http://master:8088/proxy/application_1699491849504_0003/		
239	Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill		
	job_1699491849504_0003		
240	Hadoop job information for Stage-3: number of mappers: 1; number of		
	reducers: 0		
241	2023-11-09 14:06:09,646 Stage-3 map = 0%, reduce = 0%		
242	2023-11-09 14:06:16,840 Stage-3 map = 100%, reduce = 0%, Cumulative CPU		
	2.13 sec		
243	MapReduce Total cumulative CPU time: 2 seconds 130 msec		
244	Ended Job = job_1699491849504_0003		
245	MapReduce Jobs Launched:		
246	Stage-Stage-3: Map: 1 Cumulative CPU: 2.13 sec HDFS Read: 10684 HDFS		
	Write: 1307 SUCCESS		

```
247 Total MapReduce CPU Time Spent: 2 seconds 130 msec
248 OK
249 王芝芝 100
250 张大立 92
251 张大立 91
252 刘云山 92
253 刘云山 99
254 孙庆刚 100
255 孙庆刚 98
256 易思玲 91
257 易思玲 94
258 李娜 98
259 李娜 91
260 梦媛媛 100
261 孔江涛 98
262 包小波 91
263 孙晓花 98
264 冯伟伟 98
265 冯伟伟 93
266 王小容 91
267 王小容 100
268 王小容 98
269 钱国峰 91
270 钱国峰 100
271 钱国峰 95
272 王君丹 92
273 王君丹 99
274 王凤娟 100
275 王一一 95
276 王一一 100
277 邢小雨 91
278 赵钱钱 99
279 赵钱钱 93
280 周小二 93
281 周小二 91
282 周小二 99
283 郑明山 93
284 郑明山 100
285 Time taken: 26.779 seconds, Fetched: 36 row(s)
286 hive>
287 hive> -- 查询数据, 查询各个课程及相应的选课人数
288 hive> select Cno,count(1) from sc group by Cno;
289 Query ID = root_20231109140754_ec86bbd4-f559-4fe9-803f-d76373782bab
290 Total jobs = 1
291 Launching Job 1 out of 1
292 Number of reduce tasks not specified. Estimated from input data size: 1
293 In order to change the average load for a reducer (in bytes):
294     set hive.exec.reducers.bytes.per.reducer=<number>
295 In order to limit the maximum number of reducers:
296     set hive.exec.reducers.max=<number>
297 In order to set a constant number of reducers:
298     set mapreduce.job.reduces=<number>
299 Starting Job = job_1699491849504_0004, Tracking URL =
    http://master:8088/proxy/application_1699491849504_0004/
```

```
300 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0004
301 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
302 2023-11-09 14:08:01,568 Stage-1 map = 0%, reduce = 0%
303 2023-11-09 14:08:09,757 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
1.76 sec
304 2023-11-09 14:08:15,863 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
3.64 sec
305 MapReduce Total cumulative CPU time: 3 seconds 640 msec
306 Ended Job = job_1699491849504_0004
307 MapReduce Jobs Launched:
308 Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.64 sec HDFS Read:
14115 HDFS Write: 189 SUCCESS
309 Total MapReduce CPU Time Spent: 3 seconds 640 msec
310 OK
311 1 15
312 2 15
313 3 13
314 4 16
315 5 12
316 6 11
317 Time taken: 22.935 seconds, Fetched: 6 row(s)
318 hive>
319 hive> -- 查询数据, 查询选修了3门以上的课程的学生学号
320 hive> select sno from sc group by sno having count(cno)>3;
321 Query ID = root_20231109141159_fd63bf1b-2aff-4821-816a-0bb649364ac0
322 Total jobs = 1
323 Launching Job 1 out of 1
324 Number of reduce tasks not specified. Estimated from input data size: 1
325 In order to change the average load for a reducer (in bytes):
326 set hive.exec.reducers.bytes.per.reducer=<number>
327 In order to limit the maximum number of reducers:
328 set hive.exec.reducers.max=<number>
329 In order to set a constant number of reducers:
330 set mapreduce.job.reduces=<number>
331 Starting Job = job_1699491849504_0005, Tracking URL =
http://master:8088/proxy/application_1699491849504_0005/
332 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0005
333 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
334 2023-11-09 14:12:07,476 Stage-1 map = 0%, reduce = 0%
335 2023-11-09 14:12:15,651 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
1.75 sec
336 2023-11-09 14:12:21,772 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
4.39 sec
337 MapReduce Total cumulative CPU time: 4 seconds 390 msec
338 Ended Job = job_1699491849504_0005
339 MapReduce Jobs Launched:
340 Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.39 sec HDFS Read:
14915 HDFS Write: 386 SUCCESS
341 Total MapReduce CPU Time Spent: 4 seconds 390 msec
342 OK
343 2018213201
```

```
344 2018213202
345 2018213204
346 2018213205
347 2018213206
348 2018213207
349 2018213211
350 2018213212
351 2018213213
352 2018213215
353 2018213218
354 2018213219
355 2018213222
356 Time taken: 23.127 seconds, Fetched: 13 row(s)
357 hive>
358 hive> -- 查询数据, 查询学生信息并根据学号进行升序排序
359 hive> select Sno from student order by Sno;
360 Query ID = root_20231109141356_e7a8733b-9cca-4366-9f4f-c3d3b1a64e9e
361 Total jobs = 1
362 Launching Job 1 out of 1
363 Number of reduce tasks determined at compile time: 1
364 In order to change the average load for a reducer (in bytes):
365   set hive.exec.reducers.bytes.per.reducer=<number>
366 In order to limit the maximum number of reducers:
367   set hive.exec.reducers.max=<number>
368 In order to set a constant number of reducers:
369   set mapreduce.job.reduces=<number>
370 Starting Job = job_1699491849504_0006, Tracking URL =
http://master:8088/proxy/application_1699491849504_0006/
371 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0006
372 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
373 2023-11-09 14:14:04,015 Stage-1 map = 0%, reduce = 0%
374 2023-11-09 14:14:11,289 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
1.73 sec
375 2023-11-09 14:14:17,572 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
3.59 sec
376 MapReduce Total cumulative CPU time: 3 seconds 590 msec
377 Ended Job = job_1699491849504_0006
378 MapReduce Jobs Launched:
379 Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.59 sec HDFS Read:
10624 HDFS Write: 593 SUCCESS
380 Total MapReduce CPU Time Spent: 3 seconds 590 msec
381 OK
382 2018213201
383 2018213202
384 2018213203
385 2018213204
386 2018213205
387 2018213206
388 2018213207
389 2018213208
390 2018213209
391 2018213210
392 2018213211
```

```
393 2018213212
394 2018213213
395 2018213214
396 2018213215
397 2018213216
398 2018213217
399 2018213218
400 2018213219
401 2018213220
402 2018213221
403 2018213222
404 Time taken: 23.267 seconds, Fetched: 22 row(s)
405 hive>
406 hive> -- 查询数据,按性别分区,在分区内按年龄升序排序
407 hive> set mapred.reduce.tasks=2;
408 hive> insert overwrite directory '/user/myname/stu_out'
409     > select * from student
410     > distribute by Sex sort by Sage;
411 Query ID = root_20231109143653_331e4547-4f79-4ca1-b68e-fb348d8e7b56
412 Total jobs = 1
413 Launching Job 1 out of 1
414 Number of reduce tasks not specified. Defaulting to jobconf value of: 2
415 In order to change the average load for a reducer (in bytes):
416     set hive.exec.reducers.bytes.per.reducer=<number>
417 In order to limit the maximum number of reducers:
418     set hive.exec.reducers.max=<number>
419 In order to set a constant number of reducers:
420     set mapreduce.job.reduces=<number>
421 Starting Job = job_1699491849504_0007, Tracking URL =
http://master:8088/proxy/application_1699491849504_0007/
422 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0007
423 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 2
424 2023-11-09 14:37:00,753 Stage-1 map = 0%, reduce = 0%
425 2023-11-09 14:37:08,011 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
1.82 sec
426 2023-11-09 14:37:14,130 Stage-1 map = 100%, reduce = 50%, Cumulative CPU
3.84 sec
427 2023-11-09 14:37:18,208 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
5.6 sec
428 MapReduce Total cumulative CPU time: 5 seconds 600 msec
429 Ended Job = job_1699491849504_0007
430 Moving data to directory /user/myname/stu_out
431 MapReduce Jobs Launched:
432 Stage-Stage-1: Map: 1 Reduce: 2 Cumulative CPU: 5.6 sec HDFS Read:
16825 HDFS Write: 676 SUCCESS
433 Total MapReduce CPU Time Spent: 5 seconds 600 msec
434 OK
435 Time taken: 25.769 seconds
436 hive>
437 hive> -- 插入数据
438 hive> -- 插入数据,在student表中插入一条新记录
439 hive> insert into table student values(2018213223,'王小哲','男',18,'IS');
440 Query ID = root_20231109144034_0053ca86-2fa4-4629-831e-8c7196b41375
```

```

441 Total jobs = 3
442 Launching Job 1 out of 3
443 Number of reduce tasks determined at compile time: 1
444 In order to change the average load for a reducer (in bytes):
445     set hive.exec.reducers.bytes.per.reducer=<number>
446 In order to limit the maximum number of reducers:
447     set hive.exec.reducers.max=<number>
448 In order to set a constant number of reducers:
449     set mapreduce.job.reduces=<number>
450 Starting Job = job_1699491849504_0008, Tracking URL =
http://master:8088/proxy/application_1699491849504_0008/
451 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0008
452 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
453 2023-11-09 14:40:42,419 Stage-1 map = 0%, reduce = 0%
454 2023-11-09 14:40:50,592 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
2.34 sec
455 2023-11-09 14:40:56,725 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
3.91 sec
456 MapReduce Total cumulative CPU time: 3 seconds 910 msec
457 Ended Job = job_1699491849504_0008
458 Stage-4 is selected by condition resolver.
459 Stage-3 is filtered out by condition resolver.
460 Stage-5 is filtered out by condition resolver.
461 Moving data to directory
hdfs://master:8020/user/hive_remote/warehouse/student.db/student/.hive-
staging_hive_2023-11-09_14-40-34_595_847354642008985408-1/-ext-10000
462 Loading data to table student.student
463 MapReduce Jobs Launched:
464 Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.91 sec HDFS Read:
18710 HDFS Write: 384 SUCCESS
465 Total MapReduce CPU Time Spent: 3 seconds 910 msec
466 OK
467 Time taken: 24.831 seconds
468 hive> -- 删除数据
469 hive> delete from student where Sno='2018213223';
470 FAILED: SemanticException [Error 10294]: Attempt to do update or delete
using transaction manager that does not support these operations.
471 hive>

```

注：默认hive不支持记录的删除，若要使用update和delete删除操作，需要配置hive-site.xml，这里省略。

## 任务6.5掉线率top10基站统计

自我国三大运营商获得无线牌照以来，无线用户发展迅猛，优化维护工作的好坏直接影响运营商的服务质量和满意度。而影响运营商服务质量和满意度的一个重要网络指标之一——掉话率。因此，统计分析各基站掉话率，有助于运营商做出科学决策，提升网络质量，开展优化维护工作，降低基站掉话率，才能有效的支撑业务发展，提升用户满意度。本小节的任务如下。统计出每个基站的掉话率，并按降序排序。找出掉话率比较高的前20个基站。

## 分析基本思路

根据jizhan\_information.scv文件中的信息，计算每个基站的掉话率，找出最高掉话率的基站，可以帮助运营商更好的分析高掉话率基站的具体情况，有助于安排维护人员有针对性的进行故障检测。已知基站掉话率的计算公式，如下图所示。

$$\text{掉话率} = \frac{\text{掉话总时长}}{\text{通话持续总时长}}$$

统计基站的掉话率的实现步骤如下:

1. 上传jizhan\_information.csv文件至服务器。
2. 创建myhive数据库并在myhive数据库中创建jizhan结构表。
3. 装载jizhan\_information.scv文件中的数据至jizhan表。
4. 创建jizhan\_result结果表，用以存储统计掉话率后的信息。
5. 根据掉话率公式统计各基站的掉话率，并按降序排序，找出掉话率Top20个基站。

## 任务实现

### 数据文件准备

如何本章数据已完成执行，这里可以忽略

```
1 [root@master ~]# cd /root/hadoop
2 [root@master hadoop]# hdfs dfs -put ./hadoop_data/jizhan_information.csv
  /user/myname/stu
3 [root@master hadoop]#
```

### 数据操作

在master或slave上执行hive进入hive命令行操作:

```
1 show databases;
2 use student;
3 create table jizhan(
4 record_time string,
5 imei int,
6 cell string,
7 ph_num int,
8 call_num int,
9 drop_num int,
10 duration int,
11 drop_rate double,
12 net_type string,
13 er1 int) row format delimited fields terminated by ',';
14 -- 装载数据文件
15 load data inpath '/user/myname/stu/jizhan_information.csv' into table
  jizhan;
16 -- 查询已装载的数据表内容
17 select * from jizhan limit 10;
18 -- 创建结果表
19 create table jizhan_result(
20 imei string,
21 drop_num int,
22 duration int,
```

```

23 drop_rate double);
24 -- 统计基站掉话率
25 from jizhan
26 insert into jizhan_result
27 select imei,sum(drop_num) as sdrop,sum(duration) as sdura,
28 sum(drop_num)/sum(duration) as drop_rate
29 group by imei
30 order by drop_rate desc;
31 -- 查询统计结果
32 select * from jizhan_result limit 10;

```

运行结果如:

```

1  hive> show databases;
2  OK
3  default
4  hive_remote
5  student
6  Time taken: 0.071 seconds, Fetched: 3 row(s)
7  hive> use student;
8  OK
9  Time taken: 0.036 seconds
10 hive>
11 hive> create table jizhan(
12     > record_time string,
13     > imei int,
14     > cell string,
15     > ph_num int,
16     > call_num int,
17     > drop_num int,
18     > duration int,
19     > drop_rate double,
20     > net_type string,
21     > er1 int) row format delimited fields terminated by ',';
22 OK
23 Time taken: 0.146 seconds
24 hive>
25 hive> -- 装载数据文件
26 hive> load data inpath '/user/myname/stu/jizhan_information.csv' into table
27 jizhan;
28 Loading data to table student.jizhan
29 OK
30 Time taken: 0.217 seconds
31 hive> -- 查询已装载的数据表内容
32 hive> select * from jizhan limit 10;
33 OK
34 record_time      NULL    cell    NULL    NULL    NULL    NULL    NULL
35 net_type         NULL
36 2011-07-13 00:00:00+08 356966 29448-37062 0 0 0 0
37     0.0    G    0
38 2011-07-13 00:00:00+08 352024 29448-51331 0 0 0 0
39     0.0    G    0
40 2011-07-13 00:00:00+08 353736 29448-51331 0 0 0 0
41     0.0    G    0

```

```

37 2011-07-13 00:00:00+08 353736 29448-51333 0 0 0 0
    0.0 G 0
38 2011-07-13 00:00:00+08 351545 29448-51333 0 0 0 0
    0.0 G 0
39 2011-07-13 00:00:00+08 353736 29448-51343 1 0 0 8
    0.0 G 0
40 2011-07-13 00:00:00+08 359681 29448-51462 0 0 0 0
    0.0 G 0
41 2011-07-13 00:00:00+08 354707 29448-51462 0 0 0 0
    0.0 G 0
42 2011-07-13 00:00:00+08 356137 29448-51470 0 0 0 0
    0.0 G 0
43 Time taken: 0.164 seconds, Fetched: 10 row(s)
44 hive>
45 hive> -- 统计基站掉话率
46 hive> from jizhan
47 > insert into jizhan_result
48 > select imei,sum(drop_num) as sdrop,sum(duration) as sdura,
49 > sum(drop_num)/sum(duration) as drop_rate
50 > group by imei
51 > order by drop_rate desc;
52 Query ID = root_20231109153639_a1df8724-256c-4dae-8834-fd1151de7eb3
53 Total jobs = 2
54 Launching Job 1 out of 2
55 Number of reduce tasks not specified. Defaulting to jobconf value of: 2
56 In order to change the average load for a reducer (in bytes):
57 set hive.exec.reducers.bytes.per.reducer=<number>
58 In order to limit the maximum number of reducers:
59 set hive.exec.reducers.max=<number>
60 In order to set a constant number of reducers:
61 set mapreduce.job.reduces=<number>
62 Starting Job = job_1699491849504_0009, Tracking URL =
http://master:8088/proxy/application_1699491849504_0009/
63 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0009
64 Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 2
65 2023-11-09 15:36:48,100 Stage-1 map = 0%, reduce = 0%
66 2023-11-09 15:36:56,600 Stage-1 map = 100%, reduce = 0%, Cumulative CPU
3.82 sec
67 2023-11-09 15:37:03,013 Stage-1 map = 100%, reduce = 50%, Cumulative CPU
6.58 sec
68 2023-11-09 15:37:08,368 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
9.47 sec
69 MapReduce Total cumulative CPU time: 9 seconds 470 msec
70 Ended Job = job_1699491849504_0009
71 Launching Job 2 out of 2
72 Number of reduce tasks determined at compile time: 1
73 In order to change the average load for a reducer (in bytes):
74 set hive.exec.reducers.bytes.per.reducer=<number>
75 In order to limit the maximum number of reducers:
76 set hive.exec.reducers.max=<number>
77 In order to set a constant number of reducers:
78 set mapreduce.job.reduces=<number>

```

```

79 Starting Job = job_1699491849504_0010, Tracking URL =
http://master:8088/proxy/application_1699491849504_0010/
80 Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job -kill
job_1699491849504_0010
81 Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
82 2023-11-09 15:37:21,675 Stage-2 map = 0%, reduce = 0%
83 2023-11-09 15:37:29,831 Stage-2 map = 100%, reduce = 0%, Cumulative CPU
2.01 sec
84 2023-11-09 15:37:35,982 Stage-2 map = 100%, reduce = 100%, Cumulative CPU
5.37 sec
85 MapReduce Total cumulative CPU time: 5 seconds 370 msec
86 Ended Job = job_1699491849504_0010
87 Loading data to table student.jizhan_result
88 MapReduce Jobs Launched:
89 Stage-Stage-1: Map: 1 Reduce: 2 Cumulative CPU: 9.47 sec HDFS Read:
57423621 HDFS write: 380103 SUCCESS
90 Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 5.37 sec HDFS Read:
394032 HDFS write: 264283 SUCCESS
91 Total MapReduce CPU Time Spent: 14 seconds 840 msec
92 OK
93 Time taken: 58.821 seconds
94 hive>
95 hive> -- 查询统计结果
96 hive> select * from jizhan_result limit 10;
97 OK
98 639876 1 734 0.0013623978201634877
99 356436 1 1028 9.727626459143969E-4
100 351760 1 1232 8.116883116883117E-4
101 368883 1 1448 6.906077348066298E-4
102 358849 1 1469 6.807351940095302E-4
103 358231 1 1613 6.199628022318661E-4
104 863738 2 3343 5.982650314089142E-4
105 865011 1 1864 5.36480686695279E-4
106 862242 1 1913 5.227391531625719E-4
107 350301 2 3998 5.002501250625312E-4
108 Time taken: 0.107 seconds, Fetched: 10 row(s)
109 hive>

```

## 小结

本章详细介绍了Hive的基本知识。Hive是一个构建在Hadoop之上的数据仓库工具，主要用于对存储在Hadoop文件中的数据集进行数据整理、特殊查询和分析处理。本章从Hive与传统数据库的区别出发，通过介绍Hive的基本概念，让读者需要了解Hive以及Hive架构、数据模型及其工作原理；通过介绍Hive的3种访问方式及搭建过程，让读者熟悉Hive的安装步骤和管理。接着介绍Hive的数据操作，让读者掌握HiveQL的相关操作。作为初学者，学习Hive需要实际动手操作Hive，这也是掌握Hive的关键，最后以掉线率Top20基站统计为例，详细介绍了如何使用Hive解决具体的实际问题。

## 版本历史

- Ver1.0-20231109

- 初始版本

[上一章节](#) [下一章节](#)