实验6 部署本地模式Hive和实战Hive

本实验的知识地图如图6-1所示（表示重点表示难点）。

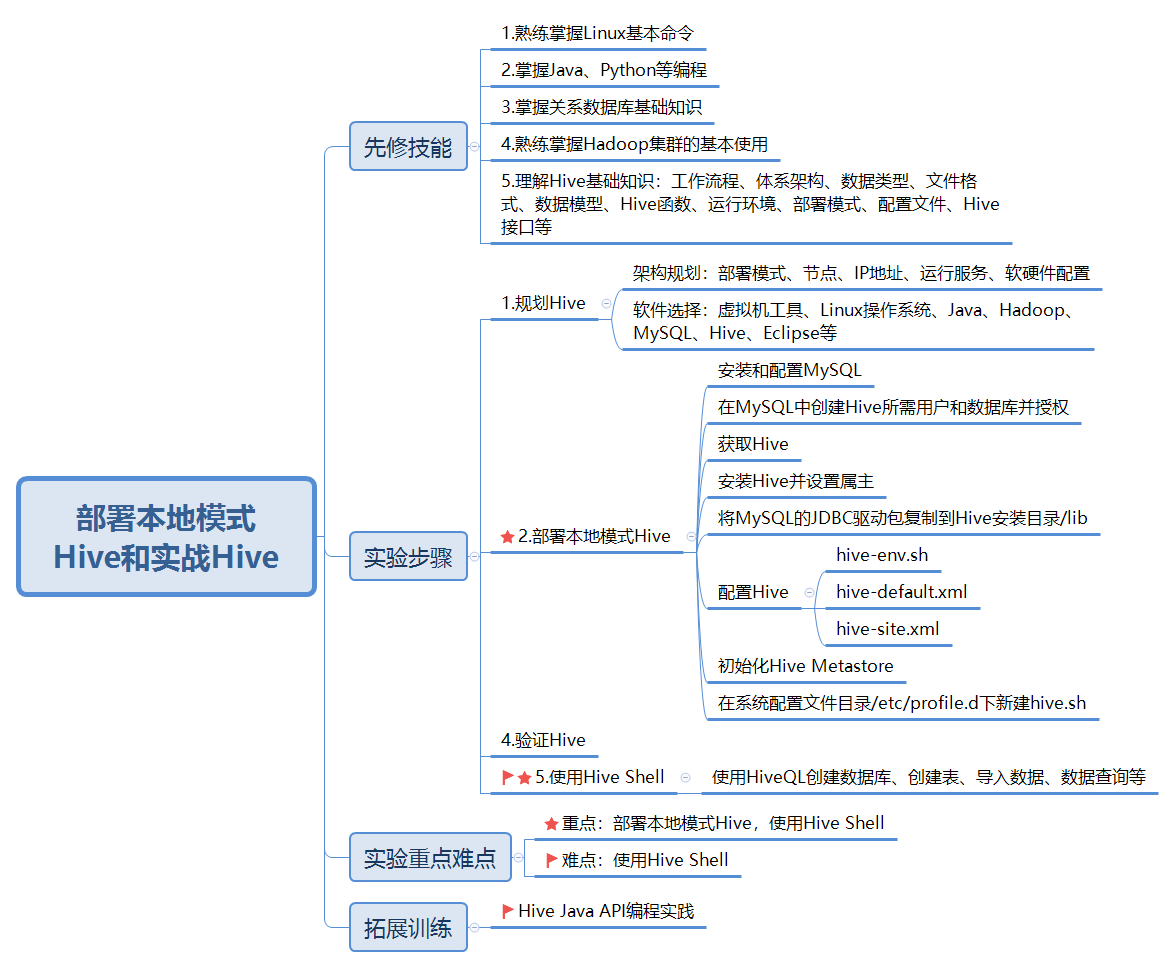


图6-1 实验6部署本地模式Hive和实战Hive知识地图

一、实验目的

1. 理解Hive工作原理。

2. 理解Hive体系架构。

3. 熟悉Hive运行模式，熟练掌握本地模式Hive的部署。

4. 了解Hive Web UI的配置和使用。

5. 熟练掌握Hive Shell常用命令的使用。

6. 了解Hive Java API，能编写简单的Hive程序。

二、实验环境

本实验所需的软件环境包括全分布模式Hadoop集群、MySQL安装包、MySQL JDBC驱动包、Hive安装包、Eclipse。

三、实验内容

1. 规划Hive。

2. 部署本地模式Hive。

3. 启动Hive。

4. 验证Hive。

5. 配置和使用Hive Web UI。

6. 使用Hive Shell常用命令。

7. 关闭Hive。

四、实验原理

（一）初识Hive

Hive由Facebook公司开源，主要用于解决海量结构化日志数据的离线分析。Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张表，并提供了类SQL查询语言HQL（Hive Query Language）。Hive的本质是将HQL语句转化成MapReduce程序，并提交到Hadoop集群上运行，其基本工作流程如图6-2所示。Hive让不熟悉MapReduce的开发人员直接编写SQL语句来实现对大规模数据的统计分析操作，大大降低了学习门槛，同时也提升了开发效率。Hive处理的数据存储在HDFS上，Hive分析数据底层的实现是MapReduce，执行程序运行在YARN上。

Hadoop集群

Hive

select city, count(1) pv from ad\_table

where data='20191001' group by city

MapReduce

YARN

HDFS

user/hive/warehouse/ad\_table/data='20191001'/log-20191001

生成MR

调度MR

运行MR读取数据

图6-2 Hive基本工作流程

Hive优点包括以下几个方面：

（1）操作接口采用类SQL语法，简单易学，提供快速开发的能力。

（2）避免编写MapReduce应用程序，减少开发人员的学习成本。

（3）Hive执行延迟比较高，常用于对实时性要求不高的海量数据分析场合中。

（4）Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。

同时，Hive也有自身缺陷，包括以下几个方面：

（1）Hive的HQL表达能力有限，例如迭代式算法无法表达，不擅长数据挖掘等。

（2）Hive效率比较低，例如Hive自动生成的MapReduce作业，通常情况下不够智能化，另外Hive粒度较粗，调优比较困难。

（二）Hive体系架构

Hive通过给用户提供的一系列交互接口，接收到用户提交的Hive脚本后，使用自身的驱动器Driver，结合元数据Metastore，将这些脚本翻译成MapReduce，并提交到Hadoop集群中执行，最后，将执行返回的结果输出到用户交互接口。Hive体系架构如图6-3所示。

Hive

Hadoop集群

MapReduce

YARN

HDFS

驱动器（Driver）

元数据

（Metastore）

CLI

Thrift Server

HWI

JDBC

ODBC

解析器（SQL Parser）

编译器（Physical Plan）

优化器（Query Optimizer）

执行器（Execution）

图6-3 Hive体系架构

由图6-3可知，Hive体系架构中主要包括如下组件：CLI、JDBC/ODBC、Thrift Server、HWI、Metastore和Driver，这些组件可以分为两类：客户端组件，服务端组件。另外，Hive还需要Hadoop的支持，它使用HDFS进行存储，使用MapReduce进行计算。

1. 客户端组件

1）CLI（Commmand Line Interface）

CLI是Hive命令行接口，是最常用的一种用户接口。CLI启动时会同时启动一个Hive副本。CLI是和Hive交互的最简单也是最常用方式，只需要在一个具备完整Hive环境下的Shell终端中键入hive即可启动服务。用户可以在CLI上输入HQL来执行创建表、更改属性以及查询等操作。不过Hive CLI不适应于高并发的生产环境，仅仅是Hive管理员的好工具。

2）JDBC/ODBC

JDBC是Java Database Connection规范，它定义了一系列Java访问各类数据库的访问接口，因此Hive-JDBC其实本质上扮演了一个协议转换的角色，把JDBC标准协议转换为访问Hive Server服务的协议。Hive-JDBC除了扮演网络协议转化的工作，并不承担其它工作，比如SQL的合法性校验和解析等。ODBC是一组对数据库访问的标准API，它的底层实现源码是采用C/C++编写的。JDBC/ODBC都是通过Hive Client与Hive Server保持通讯的，借助Thrift RPC协议来实现交互。

3）HWI（Hive Web Interface）

HWI是Hive的Web访问接口，提供了一种可以通过浏览器来访问Hive的服务。

2. 服务端组件

1）Thrift Server

Thrift是Facebook开发的一个软件框架，它用来进行可扩展且跨语言的服务开发，Hive集成了Thrift Server服务，能让不同的编程语言如Java、Python等调用Hive接口。

2）元数据（Metastore）

Metastore组件用于存储Hive的元数据，表名、表所属的数据库（默认是default）、表的拥有者、列/分区字段、表的类型（是否是外部表）、表的数据所在目录等。Hive元数据默认存储在自带的Derby数据库中，推荐使用MySQL存储Metastore。元数据对于Hive十分重要，因此Hive支持把Metastore服务独立出来，安装到远程的服务器集群里，从而解耦Hive服务和Metastore服务，保证Hive运行的健壮性。

3）驱动器（Driver）

Driver组件的作用是将用户编写的HiveQL语句进行解析、编译、优化，生成执行计划，然后调用底层的MapReduce计算框架。Hive驱动器由四部分组成：

（1）解析器（SQL Parser）：将SQL字符串转换成抽象语法树AST，这一步一般都用第三方工具库完成，例如antlr；对AST进行语法分析，例如表是否存在、字段是否存在、SQL语义是否有误。

（2）编译器（Physical Plan）：将AST编译生成逻辑执行计划。

（3）优化器（Query Optimizer）：对逻辑执行计划进行优化。

（4）执行器（Execution）：把逻辑执行计划转换成可以运行的物理计划，对于Hive来说，就是MapReduce/Spark。

这里需要说明一下Hive Server和Hive Server 2两者的联系和区别。Hive Server和Hive Server 2都是基于Thrift的，但Hive Sever有时被称为Thrift Server，而Hive Server 2却不会；两者都允许远程客户端使用多种编程语言在不启动CLI的情况下通过Hive Server和Hive Server 2对Hive中的数据进行操作。但是官方表示从Hive 0.15起就不再支持Hive Server了，为什么不再支持Hive Server了呢？这是因为Hive Server不能处理多于一个客户端的并发请求，究其原因是由于Hive Server使用Thrift接口而导致的限制，不能通过修改HiveServer的代码修正。因此在Hive 0.11.0版本中重写了Hive Server代码得到了Hive Server 2，进而解决了该问题。Hive Server 2支持多客户端的并发和认证，为开放API客户端如JDBC、ODBC提供更好的支持。

另外，还需要说明一下Hive元数据Metastore。Hive元数据是数据仓库的核心数据，完成HDFS中表数据的读写和管理功能，元数据作为一个服务进程运行。如上文所述，元数据默认存储在自带的Derby数据库中，但推荐使用关系型数据库如MySQL来存储，采用关系数据库存储元数据的根本原因在于快速响应数据存取的需求。Hive元数据通常有三种存储位置形式：嵌入式元数据、本地元数据和远程元数据，根据元数据存储位置的不同，Hive部署模式也不同，具体参考四（七）。

（三）Hive数据类型

Hive数据类型分为两类：基本数据类型和集合数据类型。

1. 基本数据类型

基本类型又称为原始类型，与大多数关系数据库中的数据类型相同。Hive的基本数据类型及说明如表6-1所示。

表6-1 Hive基本数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | | 长度 | 说明 |
| 数字类 | TINYINT | 1字节 | 有符号整型，-128 ~ 127 |
| SMALLINT | 2字节 | 有符号整型，-32768 ~ 32767 |
| INT | 4字节 | 有符号整型，-2,147,483,648 ~ 2,147,483,647 |
| BIGINT | 8字节 | 有符号整型，-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 |
| FLOAT | 4字节 | 有符号单精度浮点数 |
| DOUBLE | 8字节 | 有符号双精度浮点数 |
| DOUBLE PRECISION | 8字节 | 同DOUBLE，Hive 2.2.0开始可用 |
| DECIMAL | -- | 可带小数的精确数字字符串 |
| NUMERIC | -- | 同DECIMAL，Hive 3.0.0开始可用 |
| 日期时间类 | TIMESTAMP | -- | 时间戳，内容格式：yyyy-mm-dd hh:mm:ss[.f...] |
| DATE | -- | 日期，内容格式：YYYY­MM­DD |
| INTERVAL | -- | -- |
| 字符串类 | STRING | -- | 字符串 |
| VARCHAR | 字符数范围1 - 65535 | 长度不定字符串 |
| CHAR | 最大的字符数：255 | 长度固定字符串 |
| Misc类 | BOOLEAN | -- | 布尔类型 TRUE/FALSE |
| BINARY | -- | 字节序列 |

Hive的基本数据类型是可以进行隐式转换的，类似于Java类型转换，例如某表达式使用INT类型，TINYINT会自动转换为INT类型，但是Hive不会进行反向转换，例如，某表达式使用TINYINT类型，INT不会自动转换为TINYINT类型，它会返回错误，除非使用CAST函数。隐式类型转换规则如下所示。

（1）任何整数类型都可以隐式地转换为一个范围更广的类型，如TINYINT可以转换成INT，INT可以转换成BIGINT。

（2）所有整数类型、FLOAT和STRING类型都可以隐式地转换成DOUBLE。

（3）TINYINT、SMALLINT、INT都可以转换为FLOAT。

（4）BOOLEAN类型不可以转换为任何其它的类型。

我们可以使用CAST函数对数据类型进行显式转换，例如CAST('1' AS INT)把字符串'1'转换成整数1。如果强制类型转换失败，例如执行CAST('X' AS INT)，则表达式返回空值NULL。

2. 集合数据类型

除了基本数据类型，Hive还提供了4种集合数据类型：ARRAY、MAP、STRUCT、UNIONTYPE。所谓集合类型是指该字段可以包含多个值，有时也称复杂数据类型。Hive集合数据类型说明如表6-2所示。

表6-2 Hive集合数据类型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 长度 | 说明 |
| ARRAY | -- | 数组，存储相同类型的数据，索引从0开始，可以通过下标获取数据 |
| MAP | -- | 字典，存储键值对数据，键或者值的数据类型必须相同，通过键获取数据，MAP<primitive\_type, data\_type> |
| STRUCT | -- | 结构体，存储多种不同类型的数据，一旦生命好结构体，各字段的位置不能改变，STRUCT<col\_name : data\_type [COMMENT col\_comment], ...> |
| UNIONTYPE | -- | 联合体，UNIONTYPE<data\_type, data\_type, ...> |

（四）Hive文件格式

Hive支持多种文件格式，常用的有以下几种。

1. TEXTFILE

说明：TEXTFILE是默认文件格式，建表时用户需要指定分隔符。

存储方式：行存储。

优点：最简单的数据格式，便于和其他工具（Pig、Grep、sed、AWK）共享数据，便于查看和编辑；加载较快。

缺点：耗费存储空间，I/O性能较低；Hive不进行数据切分合并，不能进行并行操作，查询效率低。

适用场景：适用于小型查询，查看具体数据内容的测试操作。

2. SEQUENCEFILE

说明：SEQUENCEFILE是二进制键值对序列化文件格式。

存储方式：行存储。

优点：可压缩、可分割，优化磁盘利用率和I/O；可并行操作数据，查询效率高。

缺点：存储空间消耗最大；对于Hadoop生态系统之外的工具不适用，需要通过text文件转化加载。

适用场景：适用于数据量较小、大部分列的查询。

3. RCFILE

说明：RCFILE是Hive推出的一种专门面向列的数据格式，它遵循“先按列划分，再垂直划分”的设计理念。

存储方式：行列式存储。

优点：可压缩，高效的列存取；查询效率较高。

缺点：加载时性能消耗较大，需要通过text文件转化加载；读取全量数据性能低。

4. ORC

说明：ORC是优化后的RCFILE。

存储方式：行列式存储。

优缺点：优缺点与RCFILE类似，查询效率最高。

适用场景：适用于Hive中大型的存储和查询。

Hive的文件格式除了以上4种之外，还有PARQUET、AVRO等格式，各种Hive文件格式的比较如图6-4所示，从图中可以看出，在TPC-DS Scale 500数据集上做的压缩比测试中，ORC的压缩率最高。

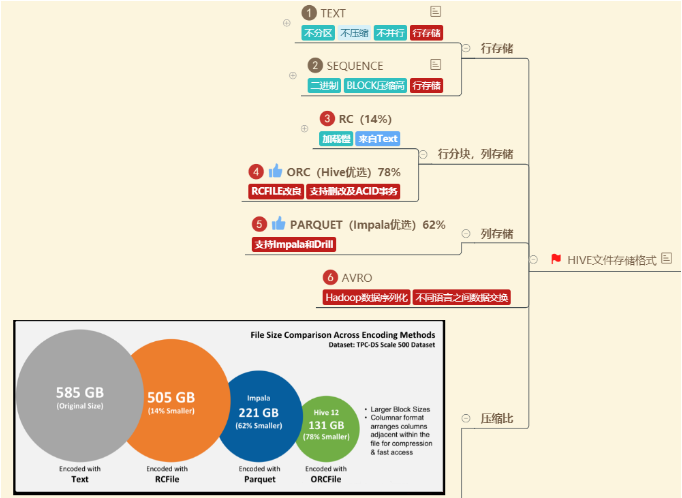


图6-4 Hive文件格式对比

（五）Hive数据模型

Hive没有专门的数据存储格式，也没有为数据建立索引，用户可以非常自由的组织Hive中的表，只需创建表时告诉Hive数据中的列分隔符和行分隔符，Hive就可以解析数据。Hive中所有的数据都存储在HDFS中，根据对数据的划分粒度，Hive包含以下数据模型：表（Table）、分区（Partition）和桶（Bucket）。如图所示，表→分区→桶，对数据的划分粒度越来越小。Hive数据模型如图6-5所示。

数据库

表

表

桶

桶

分区

分区

桶

桶

倾斜数据

正常数据

图6-5 Hive数据模型

1. 表（Table）

Hive的表和关系数据库中的表相同，具有各种关系代数操作。Hive中有两种表：内部表（Table）和外部表（External Table）。

1）内部表（Table）

Hive默认创建的表都是内部表，因为这种表，Hive会（或多或少地）控制着数据的生命周期。默认情况下Hive会将这些表的数据存储在由配置项hive.metastore.warehouse.dir（例如/user/hive/warehouse）所定义的HDFS目录的子目录下，每一个Table在该数据仓库目录下都拥有一个对应的目录存储数据。当删除一个内部表时，Hive会同时删除这个数据目录。内部表不适合和其它工具共享数据。

2）外部表（External Table）

Hive创建外部表时需要指定数据读取的目录，外部表仅记录数据所在的路径，不对数据的位置做任何改变，而内部表创建时就把数据存放到默认路径下，当删除表时，内部表会将数据和元数据全部删除，而外部表只删除元数据，数据文件不会删除。外部表和内部表在元数据的组织上是相同的，外部表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。

2. 分区（Partition）

分区表通常分为静态分区表和动态分区表，前者导入数据时需要静态指定分区，后者可以直接根据导入数据进行分区。

分区表实际上就是一个对应HDFS文件系统上的独立的文件夹，该文件夹下是该分区所有的数据文件。Hive中的分区就是分目录，把一个大的数据集根据业务需要分割成小的数据集。分区的好处是可以让数据按照区域进行分类，避免了查询时的全表扫描。

3. 桶（Bucket）

分桶就是将同一个目录下的一个文件拆分成多个文件，每个文件包含一部分数据，方便获取值，提高检索效率。

分区针对的是数据的存储路径，分桶针对的是数据文件。分区提供一个隔离数据和优化查询的便利方式，但并非所有的数据集都可形成合理的分区；分桶是将数据集分解成更容易管理的若干部分的另一种技术。

（六）Hive函数

Hive支持多种内置运算符和内置函数，方便开发人员调用。在Hive命令行中使用命令“show functions”可以查看所有函数列表，如果要查看某个函数的帮助信息，可以使用“describe function”加函数名来显示。另外，对于部分高级用户，可能需要开发自定义函数来实现特定功能。

1. 内置运算符

内置运算符包括算术运算符、关系运算符、逻辑运算符和复杂运算符，关于Hive内置运算符的说明如表6-3所示。

表6-3 Hive内置运算符

|  |  |  |
| --- | --- | --- |
| 类型 | 运算符 | 说明 |
| 算术运算符 | +、-、\*、/ | 加、减、乘、除 |
| % | 求余 |
| &、|、^、~ | 按位与、或、异或、非 |
| 关系运算符 | =、!=（或<>）、<、<=、>、>= | 等于、不等于、小于、小于等于、大于、大于等于 |
| IS NULL、IS NOT NULL | 判断值是否为“NULL” |
| LIKE、RLIKE、REGEXP | LIKE进行SQL匹配，RLIKE进行Java匹配，REGEXP与RLIKE相同 |
| 逻辑运算符 | AND、&& | 逻辑与 |
| OR、| | 逻辑或 |
| NOT、! | 逻辑非 |
| 复杂运算符 | A[n] | A是一个数组，n为int型。返回数组A的第n个元素，第一个元素的索引为0 |
| M[key] | M是Map，关键值是key，返回关键值对应的值 |
| S.x | S为struct，返回x字符串在结构S中的存储位置 |

2. 内置函数

常用内置函数包括数学函数、字符串函数、条件函数、日期函数、聚集函数、XML和JSON函数。关于Hive部分内置函数的说明如表6-4、表6-5所示。

表6-4 Hive内置函数之字符串函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| length(string A) | 返回字符串的长度 |
| reverse(string A) | 返回倒序字符串 |
| concat(string A, string B…) | 连接多个字符串，合并为一个字符串，可以接受任意数量的输入字符串 |
| concat\_ws(string SEP, string A, string B…) | 链接多个字符串，字符串之间以指定的分隔符分开 |
| substr(string A, int start) substring(string A, int start) | 从文本字符串中指定的起始位置后的字符 |
| substr(string A, int start, int len) substring(string A, int start, int len) | 从文本字符串中指定的位置指定长度的字符 |
| upper(string A) ucase(string A) | 将文本字符串转换成字母全部大写形式 |
| lower(string A) lcase(string A) | 将文本字符串转换成字母全部小写形式 |
| trim(string A) | 删除字符串两端的空格，字符之间的空格保留 |
| ltrim(string A) | 删除字符串左边的空格，其他的空格保留 |
| rtrim(string A) | 删除字符串右边的空格，其他的空格保留 |
| regexp\_replace(string A, string B, string C) | 字符串A中的B字符被C字符替换 |
| regexp\_extract(string subject, string pattern, int index) | 通过下标返回正则表达式指定的部分 |
| parse\_url(string urlString, string partToExtract [, string keyToExtract]) | 返回URL指定的部分 |
| get\_json\_object(string json\_string, string path) | select a.timestamp, get\_json\_object(a.appevents, ‘$.eventid’), get\_json\_object(a.appenvets, ‘$.eventname’) from log a; |
| space(int n) | 返回指定数量的空格 |
| repeat(string str, int n) | 重复N次字符串 |
| ascii(string str) | 返回字符串中首字符的数字值 |
| lpad(string str, int len, string pad) | 返回指定长度的字符串，给定字符串长度小于指定长度时，由指定字符从左侧填补 |
| rpad(string str, int len, string pad) | 返回指定长度的字符串，给定字符串长度小于指定长度时，由指定字符从右侧填补 |
| split(string str, string pat) | 将字符串转换为数组 |
| find\_in\_set(string str, string strList) | 返回字符串str第一次在strlist出现的位置。如果任一参数为NULL，返回NULL；如果第一个参数包含逗号，返回0 |
| sentences(string str, string lang, string locale) | 将字符串中内容按语句分组，每个单词间以逗号分隔，最后返回数组 |
| ngrams(array>, int N, int K, int pf) | SELECT ngrams(sentences(lower(tweet)), 2, 100 [, 1000]) FROM twitter; |
| context\_ngrams(array>, array, int K, int pf) | SELECT context\_ngrams(sentences(lower(tweet)), array(null,null), 100, [, 1000]) FROM twitter; |

表6-5 Hive内置函数之日期函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| from\_unixtime(bigint unixtime[, string format]) | UNIX\_TIMESTAMP参数表示返回一个值YYYY- MM- DD HH：MM：SS或YYYYMMDDHHMMSS.uuuuuu格式，这取决于是否是在一个字符串或数字语境中使用的功能。该值表示在当前的时区 |
| unix\_timestamp() | 如果不带参数的调用，返回一个Unix时间戳（从“1970- 01 - 0100:00:00”到现在的UTC秒数）为无符号整数 |
| unix\_timestamp(string date) | 指定日期参数调用UNIX\_TIMESTAMP()，它返回参数值“1970- 01 - 0100:00:00”到指定日期的秒数。 |
| unix\_timestamp(string date, string pattern) | 指定时间输入格式，返回到1970年秒数 |
| to\_date(string timestamp) | 返回时间中的年月日 |
| to\_dates(string date) | 给定一个日期date，返回一个天数（0年以来的天数） |
| year(string date) | 返回指定时间的年份，范围在1000到9999，或为“零”日期的0 |
| month(string date) | 返回指定时间的月份，范围为1至12月，或为“零”月份的0 |
| day(string date) dayofmonth(date) | 返回指定时间的日期 |
| hour(string date) | 返回指定时间的小时，范围为0到23 |
| minute(string date) | 返回指定时间的分钟，范围为0到59 |
| second(string date) | 返回指定时间的秒，范围为0到59 |
| weekofyear(string date) | 返回指定日期所在一年中的星期号，范围为0到53 |
| datediff(string enddate, string startdate) | 两个时间参数的日期之差 |
| date\_add(string startdate, int days) | 给定时间，在此基础上加上指定的时间段 |
| date\_sub(string startdate, int days) | 给定时间，在此基础上减去指定的时间段 |

读者可以使用命令“describe function <函数名>”查看该函数的英文帮助，效果如图6-6所示。

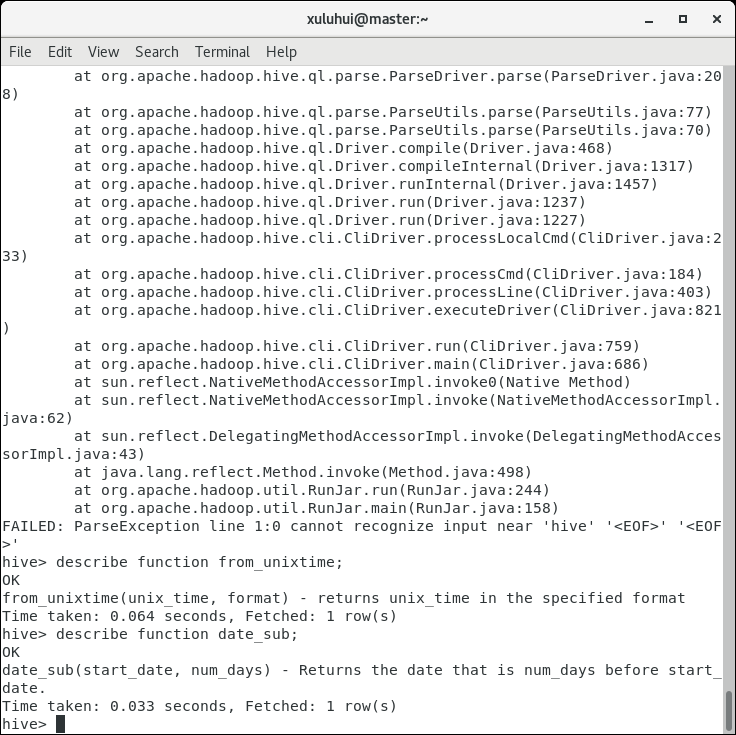


图6-6 使用命令describe function查看函数帮助

3. 自定义函数

虽然HiveQL内置了许多函数，但是在某些特殊场景下，可能还是需要自定义函数。Hive自定义函数包括三种：普通自定义函数（UDF）、表生成自定义函数（UDTF）和聚集自定义函数（UDAF）。

1）普通自定义函数（UDF）

普通UDF支持一个输入产生一个输出。普通自定义函数需要继承org.apache.hadoop.hive.ql.exec.UDF，重写类UDF中的evaluate()方法。

2）表生成自定义函数（UDTF）

表生成自定义函数UDTF支持一个输入多个输出。实现表生成自定义函数需要继承类org.apache.hadoop.hive.ql.udf.generic.GenericUDTF，需要依次实现以下三个方法：

* initialize()：行初始化，返回UDTF的输出结果的行信息（行数，类型等）。
* process()：对传入的参数进行处理，可以通过forward()返回结果。
* close()：清理资源。

3）聚集自定义函数（UDAF）

当系统自带的聚合函数不能满足用户需求时，就需要自定义聚合函数。UDAF支持多个输入一个输出。自定义聚集函数需要继承类org.apache.hadoop.hive.ql.exec.UDAF，自定义的内部类要实现接口org.apache.hadoop.hive.ql.exec.UDAFEvaluator，相对于普通自定义函数，聚集自定义函数较为复杂，需要依次实现以下五个方法：

* init()：初始化中间结果。
* iterate()：接收传入的参数，并进行内部转化，定义聚合规则，返回值为boolean类型。
* terminatePartial()：iterate结束后调用，返回当前iterate迭代结果，类似于Hadoop的Combiner。
* merge()：用于接收terminatePartial()返回的数据，进行合并操作。
* terminate()：用于返回最后聚合结果。

（七）部署Hive

1. 运行环境

对于大部分Java开源产品而言，在部署与运行之前，总是需要搭建一个合适的环境，通常包括操作系统和Java环境两方面。同时，Hive依赖于Hadoop，因此Hive部署与运行所需要的系统环境包括以下几个方面。

1）操作系统

Hive支持不同平台，在当前绝大多数主流的操作系统上都能够运行，例如Unix/Linux、Windows等。本书采用的操作系统为Linux发行版CentOS 7。

2）Java环境

Hive使用Java语言编写，因此它的运行环境需要Java环境的支持。

3）Hadoop

Hive需要Hadoop的支持，它使用HDFS进行存储，使用MapReduce进行计算。

2. 部署模式

根据元数据Metastore存储位置的不同，Hive部署模式共有以下3种。

1）内嵌模式（Embedded Metastore）

内嵌模式是Hive Metastore最简单的部署方式，使用Hive内嵌的Derby数据库来存储元数据。但是Derby只能接受一个Hive会话的访问，试图启动第二个Hive会话就会导致Metastore连接失败。Hive官方并不推荐使用内嵌模式，此模式通常用于开发者调试环境中，真正生产环境中很少使用。Hive内嵌模式示例如图6-7所示。

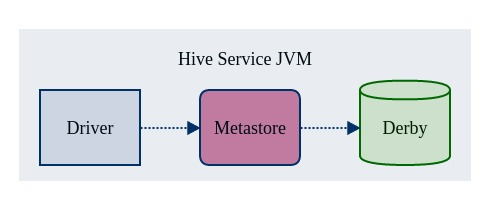


图6-7 Hive内嵌模式示例

2）本地模式（Local Metastore）

本地模式是Metastore的默认模式。该模式下，单Hive会话（一个Hive服务JVM）以组件方式调用Metastore和Driver，允许同时存在多个Hive会话，即多个用户可以同时连接到元数据库中。常见JDBC兼容的数据库如MySQL都可以使用，数据库运行在一个独立的Java虚拟机上。Hive本地模式示例如图6-8所示。

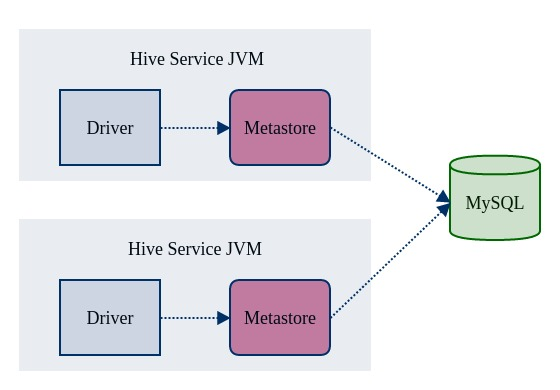


图6-8 Hive本地模式示例

3）远程模式（Remote Metastore）

远程模式将Metastore分离出来，成为一个独立的Hive服务，而不是和Hive服务运行在同一个虚拟机上。这种模式使得多个用户之间不需要共享JDBC登录帐户信息就可以存取元数据，避免了认证信息的泄漏，同时，可以部署多个Metastore服务，以提高数据仓库可用性。Hive远程模式示例如图6-9所示。

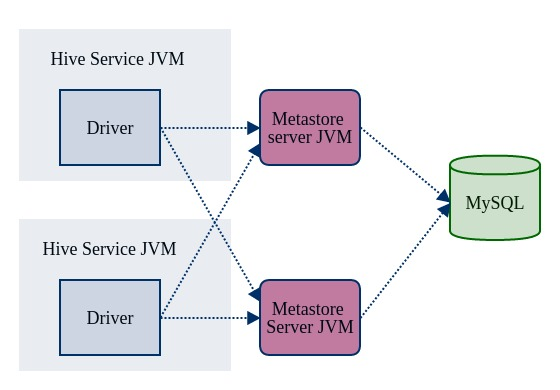


图6-9 Hive远程模式示例

3. 配置文件

Hive所有配置文件位于$HIVE\_HOME/conf下，具体的配置文件如图6-10所示。

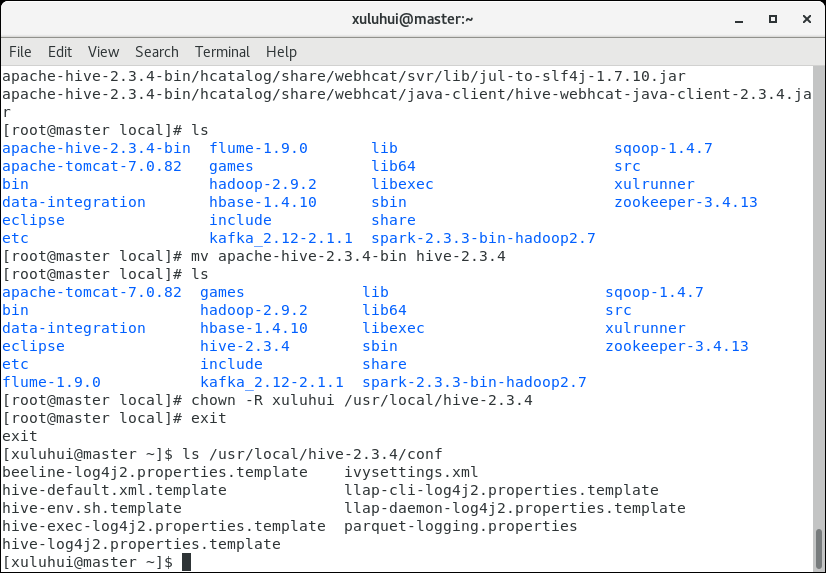


图6-10 Hive配置文件位置

用户在部署Hive时，经常编辑的配置文件有两个：hive-site.xml和hive-env.sh，它们可以在原始模板配置文件hive-env.sh.template、hive-default.xml.template的基础上创建并进行修改，另外，还需要将hive-default.xml.template复制为hive-default.xml，Hive会先加载hive-default.xml文件，再加载hive-site.xml文件，如果两个文件里有相同的配置，那么以hive-site.xml为准。Hive常用配置文件的说明如表6-7所示。

表6-7 Hive配置文件（部分）

|  |  |
| --- | --- |
| 文件名称 | 描述 |
| hive-env.sh | Bash脚本，设置Linux/Unix环境下运行Hive要用的环境变量，主要包括Hadoop安装路径HADOOP\_HOME、Hive配置文件存放路径HIVE\_CONF\_DIR、Hive运行资源库路径HIVE\_AUX\_JARS\_PATH等 |
| hive-default.xml | XML文件，Hive核心配置文件，包括Hive数据存放位置、Metastore的连接URL、JDO连接驱动类、JDO连接用户名、JDO连接密码等配置项， |
| hive-site.xml | XML文件，其配置项会覆盖默认配置hive-default.xml |

关于Hive配置参数的详细信息读者请参考官方文档https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-ConfigurationManagementOverview。其中，配置文件hive-site.xml中涉及的主要配置参数如表6-8所示。

表6-8 配置文件hive-site.xml涉及的主要参数

|  |  |
| --- | --- |
| 配置参数 | 功能 |
| hive.exec.scratchdir | HDFS路径，用于存储不同map/reduce阶段的执行计划和这些阶段的中间输出结果，默认值为/tmp/hive，对于每个连接用户，都会创建目录${hive.exec.scratchdir}/&lt;username&gt;该目录的权限为733 |
| hive.metastore.warehouse.dir | Hive默认数据文件存储路径，通常为HDFS可写路径，默认值为/user/hive/warehouse |
| hive.metastore.uris | 远程模式下Metastore的URI列表 |
| javax.jdo.option.ConnectionURL | Metastore的连接URL |
| javax.jdo.option.ConnectionDriverName | JDO连接驱动类 |
| javax.jdo.option.ConnectionUserName | JDO连接用户名 |
| javax.jdo.option.ConnectionPassword | JDO连接密码 |
| hive.hwi.war.file | HWI的war文件所在的路径 |

部署内嵌模式Hive时，配置文件hive-site.xml中需要设置的属性选项及示例如表6-9所示。

表6-9 内嵌模式Hive配置文件hive-site.xml所需配置属性示例

|  |  |
| --- | --- |
| 配置参数 | 设置值示例 |
| javax.jdo.option.ConnectionURL | jdbc:derby:;databaseName=metastore\_db;create=true |
| javax.jdo.option.ConnectionDriverName | org.apache.derby.jdbc.EmbeddedDriver |
| javax.jdo.option.ConnectionUserName | hiveEmbedded |
| javax.jdo.option.ConnectionPassword | hiveEmbedded |

部署本地模式Hive时，配置文件hive-site.xml中需要设置的属性选项及示例如表6-10所示。

表6-10 本地模式Hive配置文件hive-site.xml所需配置属性示例

|  |  |
| --- | --- |
| 配置参数 | 设置值示例 |
| javax.jdo.option.ConnectionURL | jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true&amp;useSSL=false |
| javax.jdo.option.ConnectionDriverName | com.mysql.jdbc.Driver |
| javax.jdo.option.ConnectionUserName | hiveLocal |
| javax.jdo.option.ConnectionPassword | hiveLocal |

部署远程模式Hive时，配置文件hive-site.xml中需要设置的属性选项如表6-11所示。

表6-11 远程模式Hive配置文件hive-site.xml所需配置属性示例

|  |  |
| --- | --- |
| 配置参数 | 设置值示例 |
| hive.metastore.uris | thrift://192.168.18.130:9083 |
| javax.jdo.option.ConnectionURL | jdbc:mysql://192.168.18.131:3306/hiveremote?createDatabaseIfNotExist=true&amp;useSSL=false |
| javax.jdo.option.ConnectionDriverName | com.mysql.jdbc.Driver |
| javax.jdo.option.ConnectionUserName | hiveremote |
| javax.jdo.option.ConnectionPassword | hiveremote |

（八）Hive接口

Hive用户接口主要包括三类：CLI、Client和HWI。其中，CLI（Commmand Line Interface）是Hive的命令行接口；Client是Hive的客户端，用户连接至HiveServer，在启动Client模式的时候，需要指出HiveServer所在节点，并且在该节点启动HiveServer；HWI是通过浏览器访问Hive，使用之前要启动hwi服务。

1. Hive Shell

Hive Shell命令是通过$HIVE\_HOME/bin/hive文件进行控制的，通过该文件可以进行Hive当前会话的环境管理，Hive表管理等操作。Hive命令需要使用“;”进行结束标示。在Linux终端下通过命令“hive -H”或“hive --service cli --help”可以查看帮助信息，如图6-11所示。

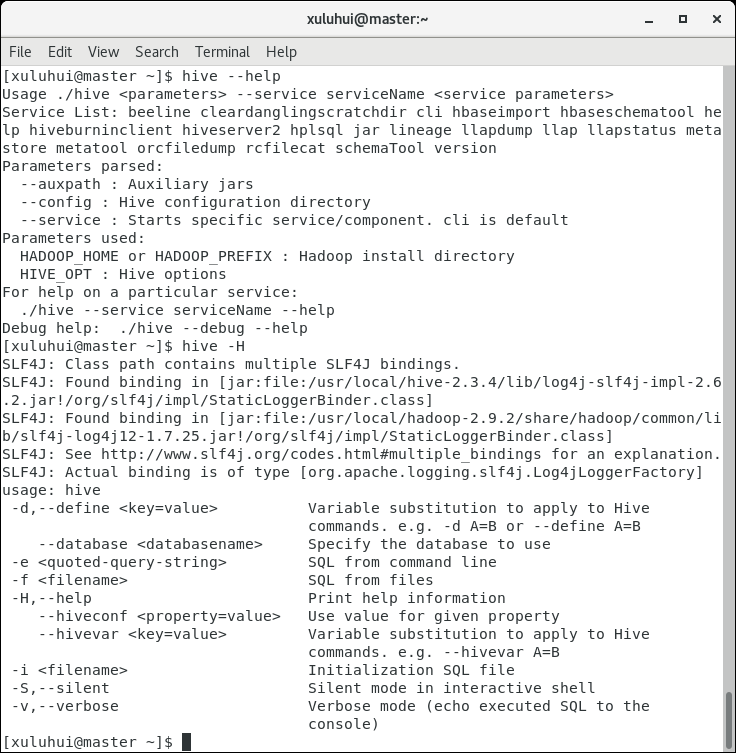


图6-11 通过命令“hive -H”查看帮助

1）Hive Shell基本命令

Hive Shell常用的基本命令主要包含退出客户端、添加文件、修改/查看环境变量、执行linux命令、执行dfs命令等。命令包括：quit、exit、set、add JAR[S] <filepath> <filepath>\*、list JAR[S]、delete JAR[S] <filepath>\*、! <linux-command>、dfs <dfs command>等。

除了Hive Shell的基本命令外，其他的命令主要是DDL、DML、select等HiveQL语句，HiveQL简称HQL，是一种类SQL的查询语言，绝大多数语法和SQL类似。

2）HiveQL

（1）HiveQL DDL

HiveQL DDL主要有数据库、表等模式的创建（CREATE）、修改（ALTER）、删除（DROP）、显示（SHOW）、描述（DESCRIBE）等命令，详细参考官方文档（网站是最新版本Hive的参考文档）https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL，HiveQL DDL具体包括的语句如图6-12所示。

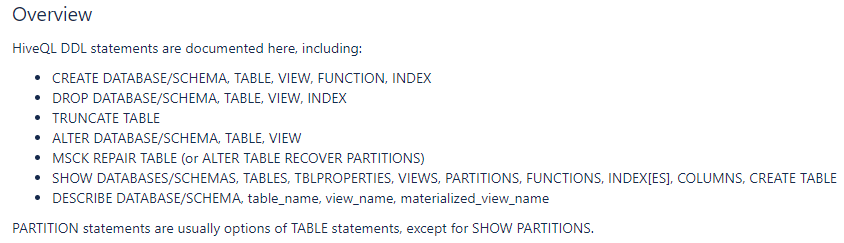


图6-12 HiveQL DDL概览

例如，创建数据库的语法如图6-13所示。

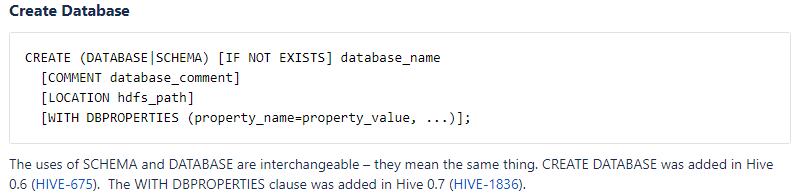


图6-13 HiveQL CREATE DATABASE 语法

例如，创建表的语法如图6-14所示。

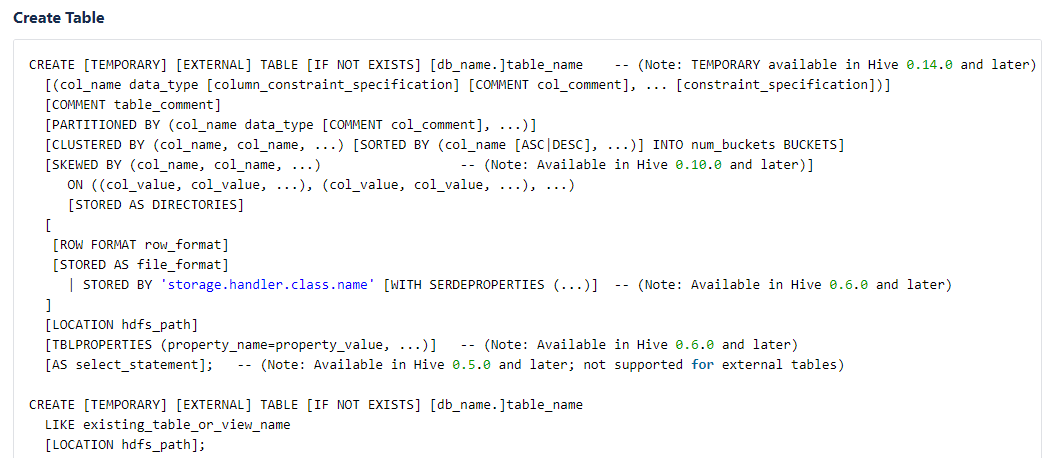


图6-14 HiveQL CREATE TABLE语法

关于创建表语法的几点说明如下：

* CREATE TABLE：创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用IF NOT EXISTS选项来忽略这个异常。
* EXTERNAL：让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION）。
* COMMENT：为表和列添加注释。
* PARTITIONED BY：创建分区表。
* CLUSTERED BY：创建分桶表。
* ROW FORMAT：指定数据切分格式。

DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]] [COLLECTION ITEMS TERMINATED BY char] [MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char] [NULL DEFINED AS char]

| SERDE serde\_name [WITH SERDEPROPERTIES (property\_name=property\_value, property\_name=property\_value, ...)]

用户在建表的时候可以自定义SerDe或者使用自带的SerDe。如果没有指定ROW FORMAT 或者ROW FORMAT DELIMITED，将会使用自带的SerDe。在建表的时候，用户还需要为表指定列，用户在指定表的列的同时也会指定自定义的SerDe，Hive通过SerDe确定表的具体列的数据。

* STORED AS：指定存储文件类型。常用的存储文件类型：SEQUENCEFILE（二进制序列文件）、TEXTFILE（文本）、RCFILE（列式存储格式文件）。
* LOCATION：指定表在HDFS上的存储位置。

（2）HiveQL DML

HiveQL DML主要有数据导入（LOAD）、数据插入（INSERT）、数据更新（UPDATE）、数据删除（DELETE）等命令，详细参考官方文档（网站是最新版本Hive的参考文档）https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML，HiveQL DDL具体包括的语句如图6-15所示。

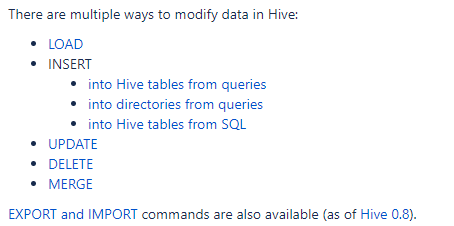


图6-15 HiveQL DML概览

（3）HiveQL SECLET

HiveQL SECLET主要有数据导入（LOAD）、数据插入（INSERT）、数据更新（UPDATE）、数据删除（DELETE）等命令，详细参考官方文档（网站是最新版本Hive的参考文档https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Select），HiveQL SECLET具体语法如图6-16所示。

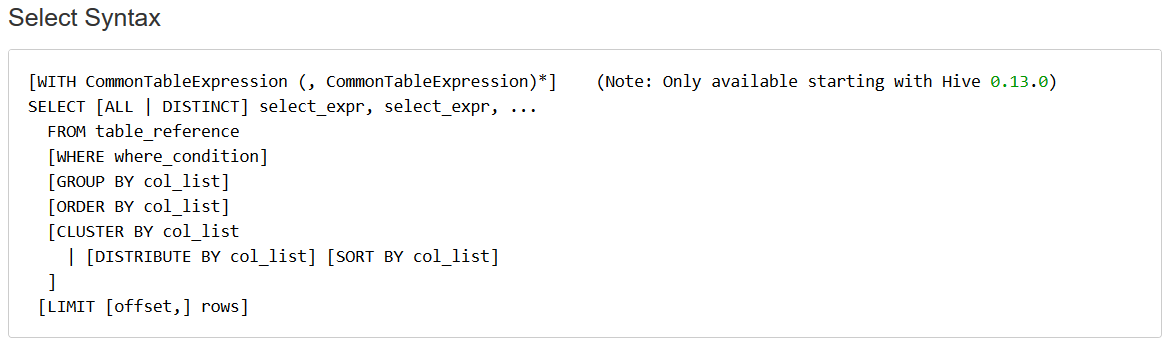


图6-16 HiveQL SELECT语法

2. Hive Web Interface（HWI）

Hive Web Interface（HWI）是Hive自带的一个Web-GUI，功能不多，可用于效果展示。由于Hive的bin目录中没有包含HWI的页面，因此需要首先下载源码从中提取jsp文件并打包成war文件到Hive安装目录下lib目录中；然后编辑配置文件hive-site.xml，添加属性参数“hive.hwi.war.file”的配置；这时在浏览器中输入<IP>:9999/hwi会出现错误“JSP support not configured”以及后续的“Unable to find a javac compiler”，究其原因，需要以下4个jar包：commons-el.jar、jasper-compiler-X.X.XX.jar、jasper-runtime-X.X.XX.jar、jdk下的tools.jar，将这些jar包拷贝到Hive的lib目录下；最后使用命令“hive --service hwi”启动HWI，在浏览器中输入<IP>:9999/hwi即可看到Hive Web页面。

3. Hive API

Hive支持Java、Python等语言编写的JDBC/ODBC应用程序访问Hive，Hive API详细参考官方文档http://hive.apache.org/javadocs/，其中有各种版本的Hive Java API，例如Hive 2.3.6 API如图6-17所示。

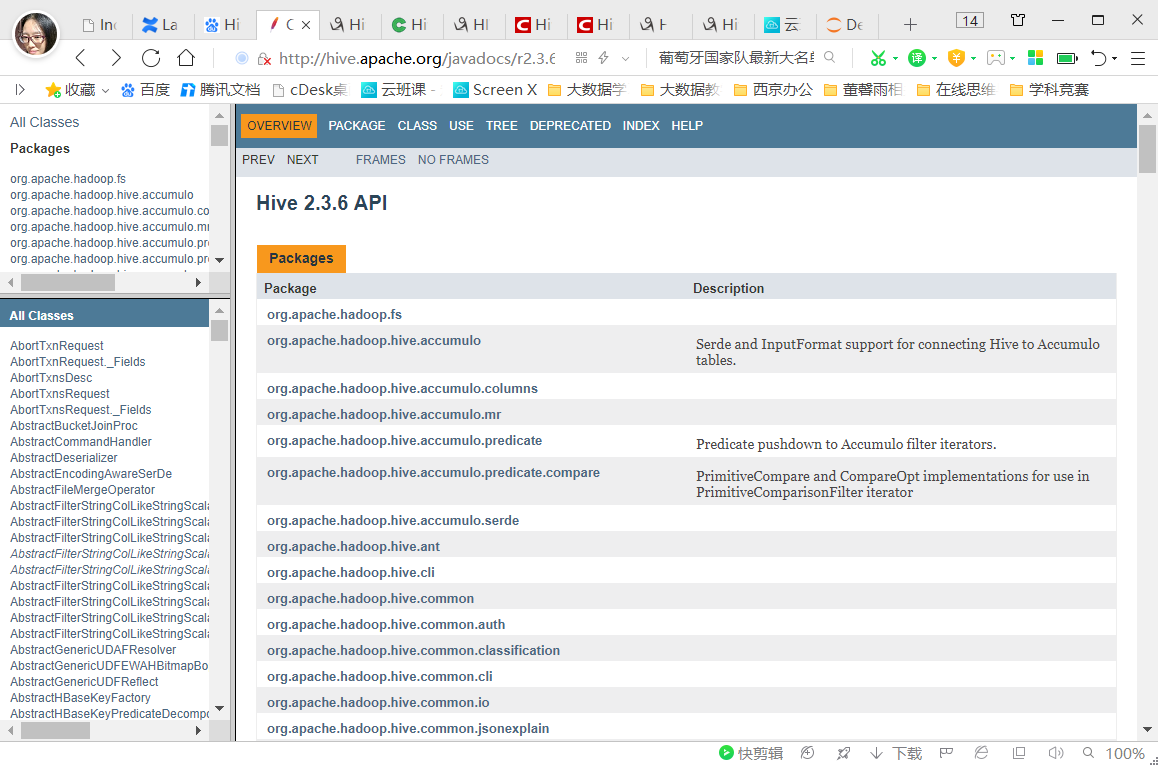


图6-17 Hive 2.3.6 API官方参考指南首页

五、实验步骤

（一）规划Hive

1. 部署规划

本实验拟部署本地模式Hive，使用MySQL存储元数据Metastore，使用全分布模式Hadoop集群。本实验使用3台安装有Linux操作系统的机器，主机名分别为master、slave1、slave2，将Hive和MySQL部署在master（192.168.18.131）节点上，全分布模式Hadoop集群部署在3个节点上。具体Hive部署规划表如表6-12所示。

表6-12 本地模式Hive部署规划表

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | IP地址 | 运行服务 | 软硬件配置 |
| master | 192.168.18.130 | NameNode  SecondaryNameNode  ResourceManager  JobHistoryServer  MySQL  Hive | 内存：4G  CPU：1个2核  硬盘：40G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2  MySQL：MySQL 5.7.27  Hive：Hive 2.3.4  Eclipse：Eclipse IDE 2018-09 for Java Developers |
| slave1 | 192.168.18.131 | DataNode  NodeManager | 内存：1G  CPU：1个1核  硬盘：20G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2 |
| slave2 | 192.168.18.132 | DataNode  NodeManager | 内存：1G  CPU：1个1核  硬盘：20G  操作系统：CentOS 7.6.1810  Java：Oracle JDK 8u191  Hadoop：Hadoop 2.9.2 |

2. 软件选择

本实验中所使用各种软件的名称、版本、发布日期及下载地址如表6-13所示。

表6-13 本实验使用的软件名称、版本、发布日期及下载地址

|  |  |  |  |
| --- | --- | --- | --- |
| 软件名称 | 软件版本 | 发布日期 | 下载地址 |
| VMware Workstation Pro | VMware Workstation 14.5.7 Pro for Windows | 2017年6月22日 | https://www.vmware.com/products/workstation-pro.html |
| CentOS | CentOS 7.6.1810 | 2018年11月26日 | https://www.centos.org/download/ |
| Java | Oracle JDK 8u191 | 2018年10月16日 | http://www.oracle.com/technetwork/java/javase/downloads/index.html |
| Hadoop | Hadoop 2.9.2 | 2018年11月19日 | http://hadoop.apache.org/releases.html |
| MySQL Connector/J | MySQL Connector/J 5.1.48 | 2019年7月29日 | https://dev.mysql.com/downloads/connector/j/ |
| MySQL Community Server | MySQL Community 5.7.27 | 2019年7月22日 | http://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm |
| Hive | Hive 2.3.4 | 2018年11月7日 | https://hive.apache.org/downloads.html |
| Eclipse | Eclipse IDE 2018-09 for Java Developers | 2018年9月 | https://www.eclipse.org/downloads/packages/ |

注意，本书采用的是Hive版本是2.3.4，3个节点的机器名分别为master、slave1、slave2，IP地址依次为192.168.18.130、192.168.18.131、192.168.18.132，后续内容均在表6-12规划基础上完成，请读者务必确认自己的Hive版本、机器名等信息。

（二）部署本地模式Hive

Hive目前1.x、2.x、3.x三个系列版本，建议读者使用当前的稳定版本，本书采用稳定版本Hive 2.3.4，因此本章的讲解都是针对这个版本进行的。尽管如此，由于Hive各个版本在部署和运行方式上的变化不大，因此本章的大部分内容都适用于Hive其他版本。

1. 初始软硬件环境准备

（1）准备3台机器，安装操作系统，编者使用CentOS Linux 7。

（2）对集群内每一台机器，配置静态IP、修改机器名、添加集群级别域名映射、关闭防火墙。

（3）对集群内每一台机器，安装和配置Java，要求Java 1.7或更高版本，编者使用Oracle JDK 8u191。

（4）安装和配置Linux集群中主节点到从节点的SSH免密登录。

（5）在Linux集群上部署全分布模式Hadoop集群，编者采用Hadoop 2.9.2。

以上步骤编者已在教材实验1中详细介绍，具体操作过程请读者参见教材，此处不再赘述，本实验从MySQL开始讲述。

2. 安装和配置MySQL

MySQL在Linux下提供多种安装方式，例如二进制方式、源码编译方式、YUM方式等，其中YUM方式比较简便，但需要网速的支持。编者采用YUM方式安装MySQL 5.7。

1）下载MySQL官方的Yum Repository

CentOS 7不支持MySQL，其Yum源中默认没有MySQL，为了解决这个问题，需要先下载MySQL的Yum Repository。读者可以直接使用浏览器到下载链接http://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm下进行下载，或者使用命令wget完成，使用命令如下所示，假设当前目录是“/home/xuluhui/Downloads”，下载到该目录下。

wget http://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

该命令运行效果如图6-18所示。



图6-18 使用wget下载MySQL的Yum Repository

2）安装MySQL官方的Yum Repository

使用以下命令安装MySQL官方的Yum Repository，使用命令如下所示，运行效果如图6-19所示。

rpm -ivh mysql57-community-release-el7-11.noarch.rpm

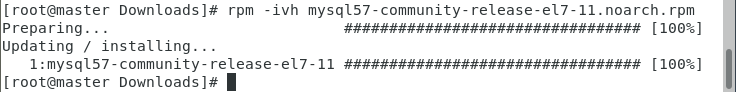


图6-19 使用rpm安装MySQL的Yum Repository

安装完这个包后，会获得两个MySQL的yum repo源：/etc/yum.repos.d/mysql-community.repo和/etc/yum.repos.d/mysql-community-source.repo。

也可以通过以下命令检查MySQL的yum repo源是否安装成功，使用命令如下所示，运行效果如图6-20所示，看到图6-20即表示安装成功。

yum repolist enabled | grep "mysql.\*-community.\*"

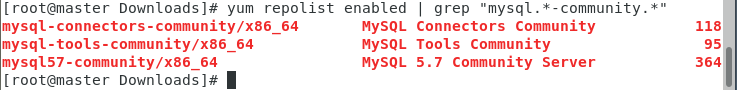


图6-20 使用yum repolist检查MySQL的yum repo源是否安装成功

3）查看提供的MySQL版本

使用以下命令查看有哪些版本的MySQL，命令运行效果如图6-21所示，从图6-21可以看出，MySQL 5.5、5.6、5.7、8.0均有。

yum repolist all | grep mysql

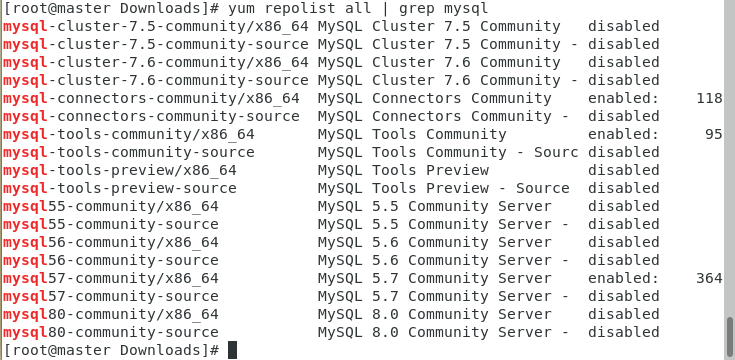


图6-21 使用yum repolist查看有哪些版本MySQL

4）安装MySQL

编者采用默认的MySQL 5.7进行安装，使用以下命令安装mysql-community-server，其他相关的依赖库mysql-community-client、mysql-community-common和mysql-community-libs均会自动安装。使用命令如下所示，运行效果如图6-22所示。

yum install -y mysql-community-server

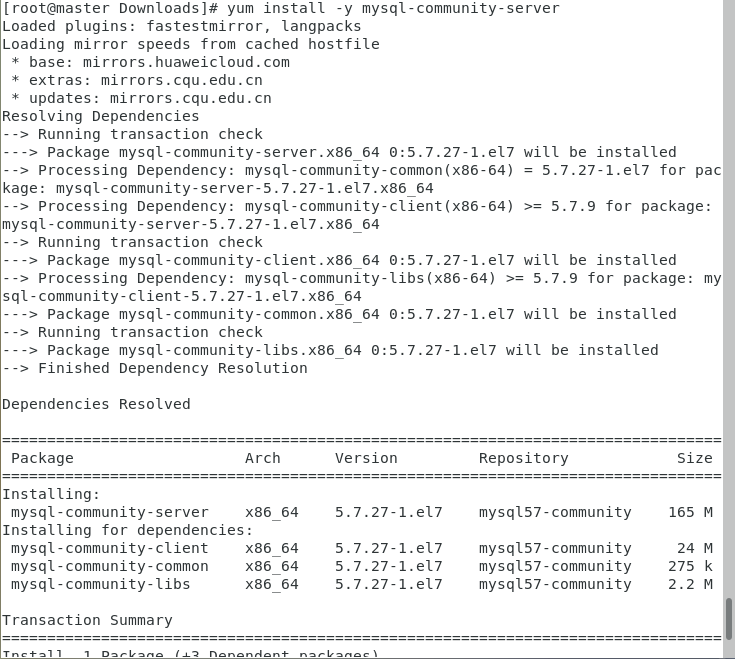


图6-22 使用yum repolist安装MySQL

当看到“Complete!”提示后，MySQL就安装完成了，接下来启动MySQL并进行登录数据库的测试。

5）启动MySQL

使用以下命令启动MySQL。读者请注意，CentOS 6中，使用service mysqld start启动MySQL。

systemctl start mysqld

还可以使用命令“systemctl status mysqld”查看状态，命令运行效果如图6-23所示，从图6-23中可以看出，MySQL已经启动了。

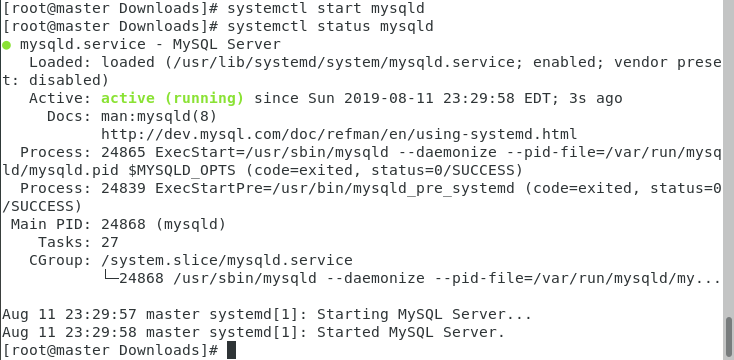


图6-23 启动MySQL和查看状态

6）测试MySQL

（1）使用root和空密码登录测试

使用root用户和空密码登录数据库服务器，使用的命令如下所示，效果如图6-24所示。

mysql -u root -p

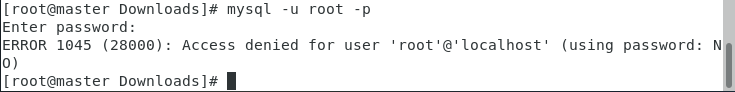


图6-24 第一次启动MySQL后使用root和空密码登录

从图\*-\*中看出，系统报错，这是因为MySQL 5.7调整了策略，新安装数据库之后，默认root密码不是空的了。在启动时随机生成了一个密码，可以/var/log/mysqld.log中找到临时密码，使用命令“grep 'temporary password' /var/log/mysqld.log”查找临时密码，效果如图6-25所示。

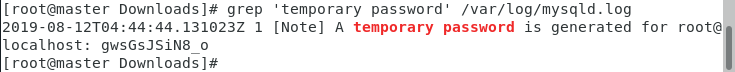


图6-25 使用grep命令查看root的初始临时密码

（2）使用root和初始化临时密码登录测试

使用root和其临时密码再次登录数据库，此时可以成功登录，但是不能做任何事情，如图6-26所示，输入命令“show databases;”显示出错信息“ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.”，这是因为MySQL 5.7默认必须修改密码之后才能操作数据库。

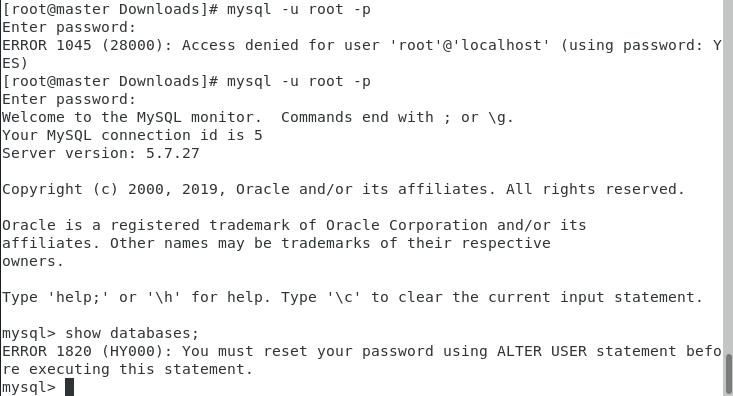


图6-26 第一次启动MySQL后使用root和初始临时密码登录

（3）修改root的初始化临时密码

在MySQL下使用如下命令修改root密码，例如新密码为“xijing”，执行效果如图6-27所示。

ALTER USER 'root'@'localhost' IDENTIFIED BY 'xijing';

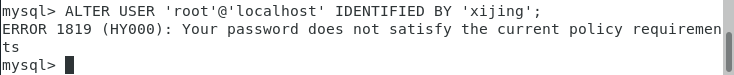


图6-27 修改root的初始化临时密码失败

从图6-27中可以看出，系统提示错误“ERROR 1819 (HY000): Your password does not satisfy the current policy requirements”，这是由于MySQL 5.7默认安装了密码安全检查插件（validate\_password），默认密码检查策略要求密码必须包含：大小写字母、数字和特殊符号，并且长度不能少于8位。读者若按此密码策略修改root密码成功后，可以使用如下命令通过MySQL环境变量查看默认密码策略的相关信息，命令运行效果如图6-28所示。

show variables like '%password%';

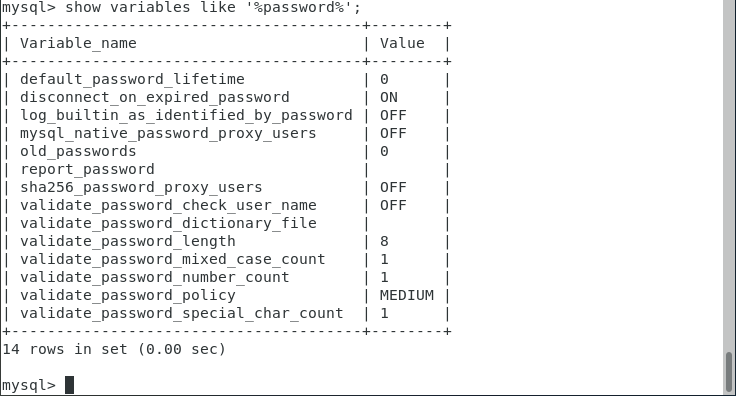


图6-28 MySQL 5.7默认密码策略

关于MySQL密码策略中部分常用相关参数的说明如表6-14所示。

表6-14 MySQL密码策略中相关参数说明（部分）

|  |  |
| --- | --- |
| 参数 | 说明 |
| validate\_password\_dictionary\_file | 指定密码验证的密码字典文件路径 |
| validate\_password\_length | 固定密码的总长度，默认为8，至少为4 |
| validate\_password\_mixed\_case\_count | 整个密码中至少要包含大/小写字母的个数，默认为1 |
| validate\_password\_number\_count | 整个密码中至少要包含阿拉伯数字的个数，默认为1 |
| validate\_password\_special\_char\_count | 整个密码中至少要包含特殊字符的个数，默认为1 |
| validate\_password\_policy | 指定密码的强度验证等级，默认为MEDIUM。  validate\_password\_policy的取值有3种：   * 0/LOW：只验证长度 * 1/MEDIUM：验证长度、数字、大小写、特殊字符 * 2/STRONG：验证长度、数字、大小写、特殊字符、字典文件 |

读者可以通过修改密码策略使密码“xijing”有效，步骤如下：

① 设置密码的验证强度等级“validate\_password\_policy”为“LOW”，注意选择“STRONG”时需要提供密码字典文件。方法是：修改配置文件/etc/my.cnf，在最后添加“validate\_password\_policy”配置，指定密码策略，为了使密码“xijing”有效，编者选择“LOW”，具体内容如下所示。

validate\_password\_policy=LOW

② 设置密码长度“validate\_password\_length”为“6”，注意密码长度最少为4。方法是：继续修改配置文件/etc/my.cnf，在最后添加“validate\_password\_length”配置，具体内容如下所示。

validate\_password\_length=6

③ 保存配置/etc/my.cnf并退出，重新启动MySQL服务使配置生效，使用的命令如下所示。

systemctl restart mysqld

（4）再次修改root的初始化临时密码

使用root和初始化临时密码登录MySQL，再次修改root密码，如新密码为“xijing”，执行效果如图6-29所示，从图6-29中可以看出，本次修改成功，密码“xijing”符合当前的密码策略。

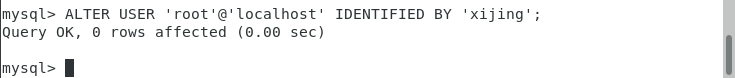


图6-29 修改MySQL密码策略后修改root密码为“xijing”成功

使用命令“flush privileges;”刷新MySQL的系统权限相关表。

（5）使用root和新密码登录测试

使用root和新密码“xijing”登录MySQL，效果如图6-30所示，从图6-30中可以看出，成功登录且可以使用命令“show databases;”。

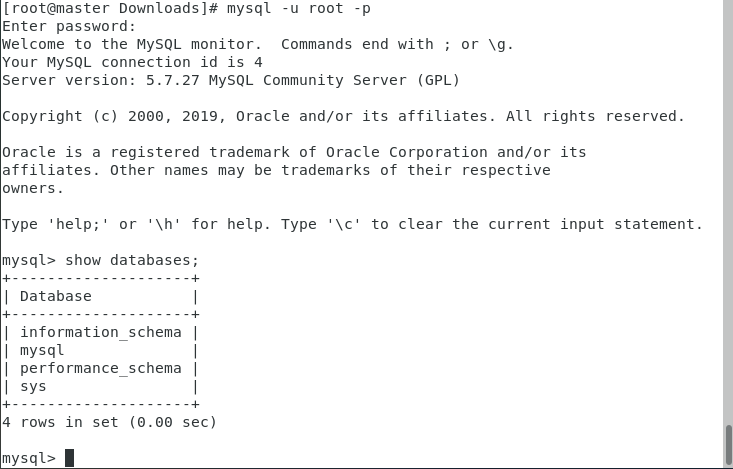


图6-30 使用root和新密码登录测试成功

3. 在MySQL中创建Hive所需用户和数据库并授权

编者计划Hive配置文件hive-site.xml中用到MySQL用户hive、密码xijing，并对数据库hive拥有所有权限。所以本步骤将带领读者在MySQL中创建用户hive、创建数据库hive、并授予数据库hive的所有权限给用户xijing。

（1）在MySQL中创建用户hive，密码为hive，使用的命令及运行效果如图6-31所示。

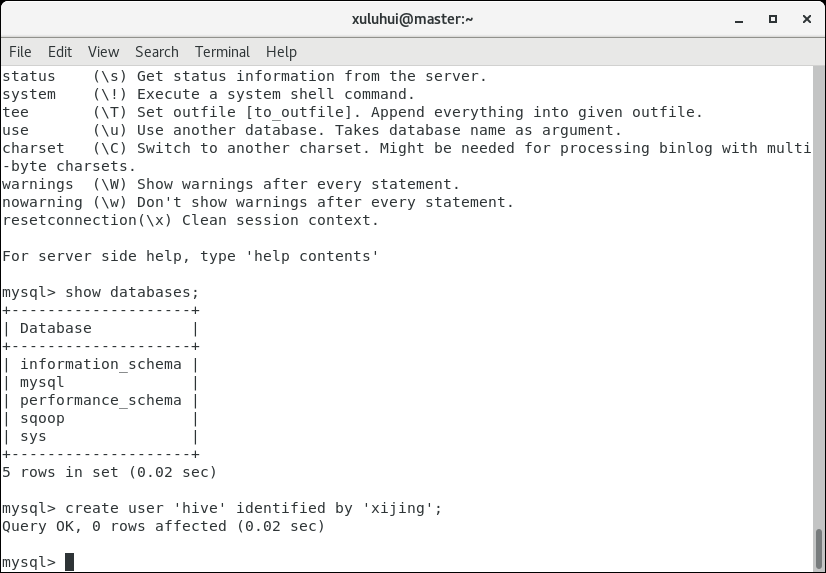


图6-31 创建MySQL用户hive

（2）创建数据库hive，使用的命令及运行效果如图6-32所示。

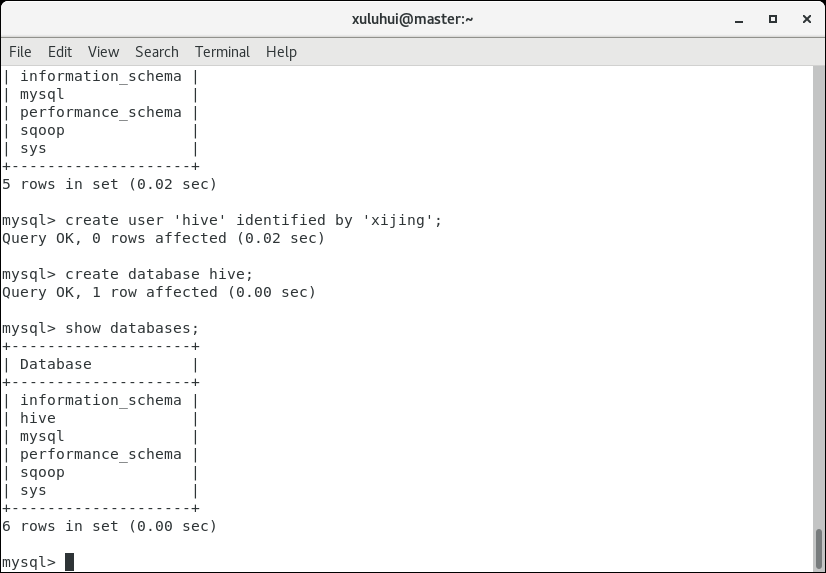


图6-32 创建并查看MySQL数据库hive

（3）将数据库hive的所有权限授权于用户hive，使用的命令及运行效果如图6-33所示。

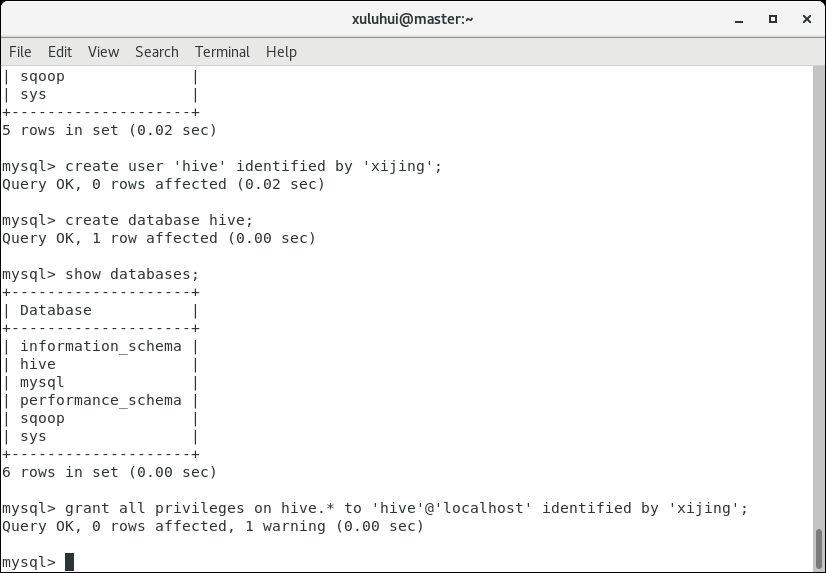


图6-33 授予数据库hive所有权限给用户hive

（4）刷新权限，使其立即生效，使用的命令及运行效果如图6-34所示。

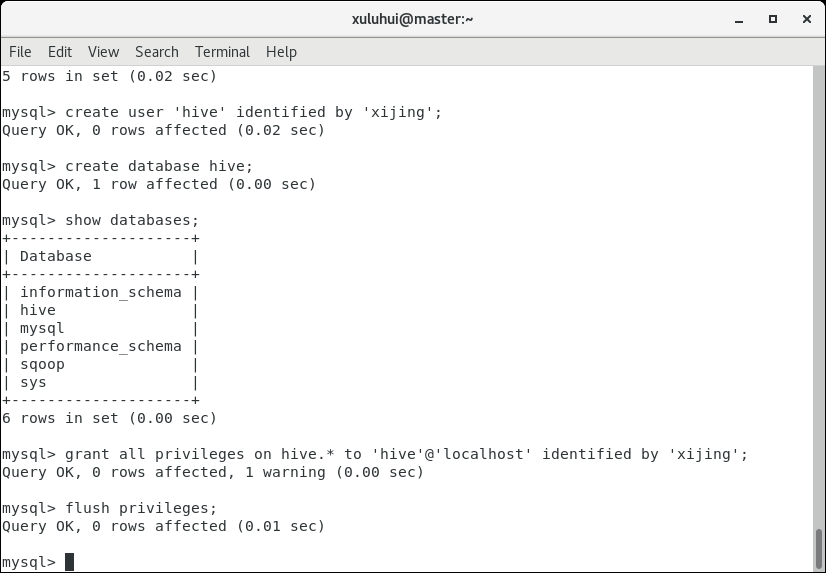


图6-34 刷新权限

（5）使用hive用户登录，并查看是否能看到数据库hive，使用的命令及运行效果如图6-35所示，从图6-35中可以看出，hive用户可以成功看到数据库hive。

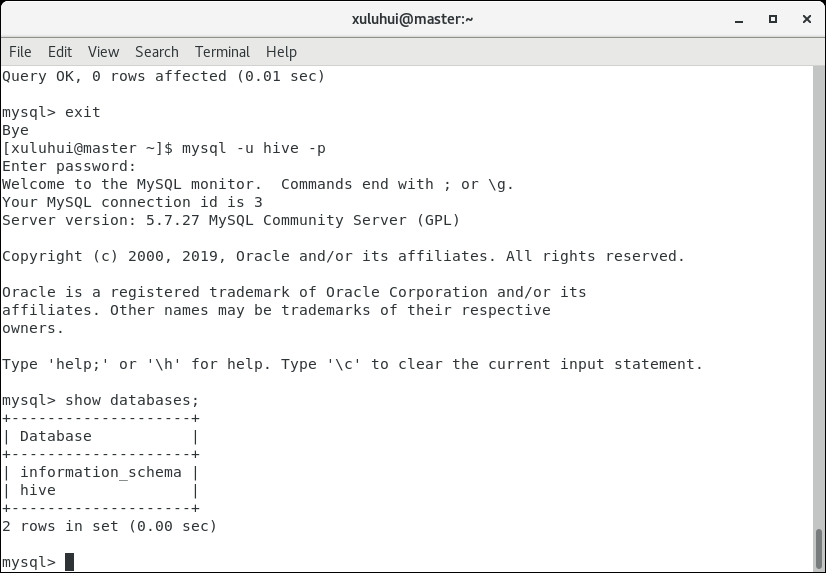


图6-35 使用hive用户登录

4. 获取Hive

Hive官方下载地址为https://hive.apache.org/downloads.html，建议读者下载stable目录下的当前稳定版本。编者采用的Hive稳定版本是2018年11月7日发布的Hive 2.3.4，其安装包文件apache-hive-2.3.4-bin.tar.gz例如存放在master机器的/home/xuluhui/Downloads中。

5. 安装Hive并设置属主

（1）在master机器上，切换到root，解压apache-hive-2.3.4-bin.tar.gz到安装目录如/usr/local下，依次使用的命令如下所示。

su root

cd /usr/local

tar -zxvf /home/xuluhui/Downloads/apache-hive-2.3.4-bin.tar.gz

（2）由于Hive的安装目录名字过长，可以使用mv命令将安装目录重命名为hive-2.3.4，使用以下命令完成。此步骤可以省略，但下文配置时Hive的安装目录就是“apache-hive-2.3.4-bin”。

mv apache-hive-2.3.4-bin hive-2.3.4

（3）为了在普通用户下使用Hive，将Hive安装目录的属主设置为Linux普通用户例如xuluhui，使用以下命令完成。

chown -R xuluhui /usr/local/hive-2.3.4

6. 将MySQL的JDBC驱动包复制到Hive安装目录/lib

（1）获取MySQL的JDBC驱动包，并保存至/home/xlh/Downloads下，下载地址为https://dev.mysql.com/downloads/connector/j/。编者使用的版本是2019年7月29日发布的MySQL Connector/J 5.1.48，文件名是mysql-connector-java-5.1.48.tar.gz

（2）将mysql-connector-java-5.1.48.tar.gz解压至如/home/xlh/Downloads下，使用的命令如下所示。

cd /home/xlh/Downloads

tar -zxvf /home/xuluhui/Downloads/mysql-connector-java-5.1.48.tar.gz

（3）将解压文件下的MySQL JDBC驱动包mysql-connector-java-5.1.48-bin.jar移动至Hive安装目录/usr/local/hive-2.3.4/lib下，并删除目录mysql-connector-java-5.1.41，依次使用的命令如下所示。

mv mysql-connector-java-5.1.48/mysql-connector-java-5.1.48-bin.jar /usr/local/hive-2.3.4/lib

rm -rf mysql-connector-java-5.1.48

7. 配置Hive

Hive所有配置文件位于$HIVE\_HOME/conf下，具体的配置文件如前文图6-9所示。本实验中编者在原始模板配置文件hive-env.sh.template、hive-default.xml.template的基础上创建并配置hive-env.sh、hive-site.xml两个配置文件。

假设当前目录为“/usr/local/hive-1.4.10/conf”，切换到普通用户如xuluhui下，在主节点master上配置Hive的具体过程如下所示。

1）配置文件hive-env.sh

环境配置文件hive-env.sh用于指定Hive运行时的各种参数，主要包括Hadoop安装路径HADOOP\_HOME、Hive配置文件存放路径HIVE\_CONF\_DIR、Hive运行资源库路径HIVE\_AUX\_JARS\_PATH等。

（1）使用命令“cp hive-env.sh.template hive-env.sh”复制模板配置文件hive-env.sh.template并命名为“hive-env.sh”。

（2）使用命令“vim hive-env.sh”编辑配置文件hive-env.sh。

① 将第48行HADOOP\_HOME注释去掉，并指定为个人机器上的Hadoop安装路径，编者修改后的内容如图6-36所示。

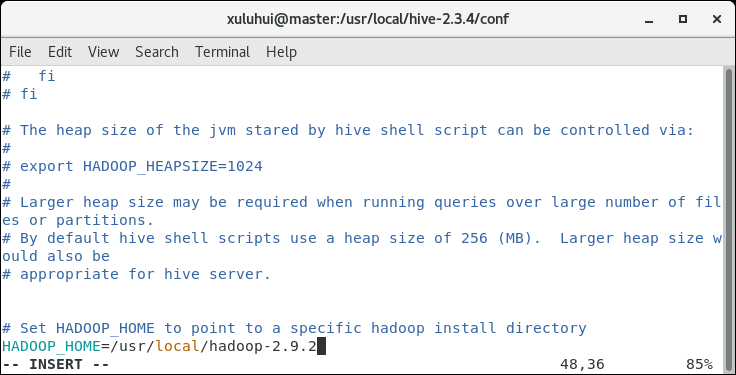


图6-36 配置HADOOP\_HOME

② 将第51行HIVE\_CONF\_DIR注释去掉，并指定为个人机器上的Hive配置文件存放路径，编者修改后的内容如图6-37所示。

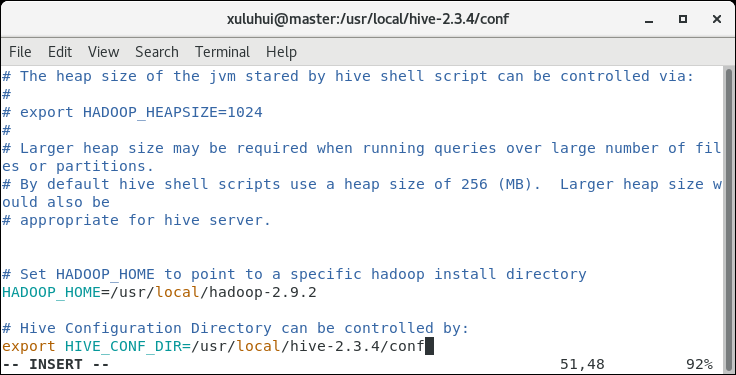


图6-37 配置HIVE\_CONF\_DIR

② 将第51行HIVE\_AUX\_JARS\_PATH注释去掉，并指定为个人机器上的Hive运行资源库路径，编者修改后的内容如图6-38所示。

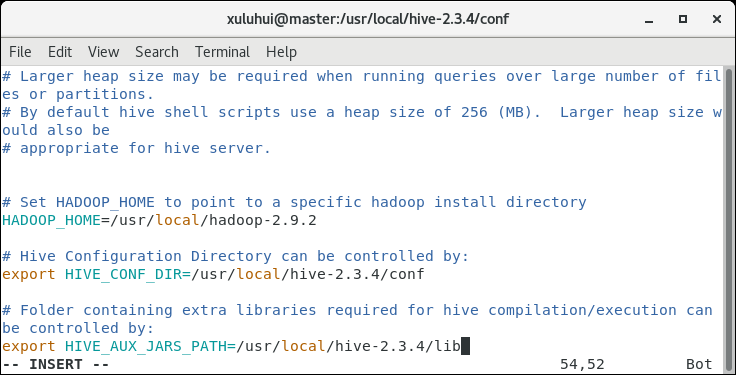


图6-38 配置HIVE\_AUX\_JARS\_PATH

2）配置文件hive-default.xml

使用命令“cp hive-default.xml.template hive-default.xml”复制模板配置文件为hive-default.xml，这是Hive默认加载的文件。

3）配置文件hive-site.xml

新建hive-site.xml，写入MySQL的配置信息。读者请注意，此处不必复制配置文件模板“hive-default.xml.template”为“hive-site.xml”，模板中参数过多，不宜读。hive-site.xml中添加的内容如下所示。

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true&amp;useSSL=false</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hive</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>xijing</value>

</property>

</configuration>

8. 初始化Hive Metastore

此时，启动Hive CLI，若输入Hive Shell命令例如“show databases;”，会出现错误，如图6-39所示，图6-39告知不能初始化Hive Metastore。

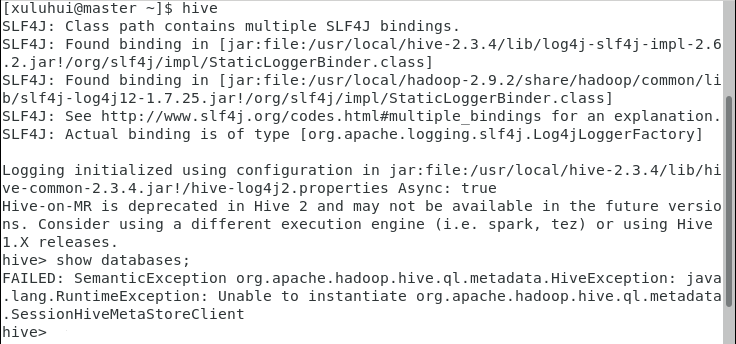


图6-39 未初始化启动Hive CLI出错

解决方法是使用命令“schemaTool -initSchema -dbType mysql”初始化元数据，将元数据写入MySQL中，执行效果如图6-40所示，若出现信息“schemaTool completed”，即表示初始化成功。



图6-40 使用命令schemaTool初始化元数据

至此，本地模式Hive已安装和配置完毕。

9. 在系统配置文件目录/etc/profile.d下新建hive.sh

另外，为了方便使用Hive各种命令，可以在Hive所安装的机器上使用“vim /etc/profile.d/hive.sh”命令在/etc/profile.d文件夹下新建文件hive.sh，并添加如下内容。

export HIVE\_HOME=/usr/local/hive-2.3.4

export PATH=$HIVE\_HOME/bin:$PATH

重启机器，使之生效。

此步骤可省略，之所以将$HIVE\_HOME/bin目录加入到系统环境变量PATH中，是因为当输入启动和管理Hive命令时，无需再切换到$HIVE\_HOME/bin目录，否则会出现错误信息“bash: \*\*\*\*: command not found...”。

（三）验证Hive

1. 启动Hadoop集群

启动全分布模式Hadoop集群的守护进程，只需在主节点master上依次执行以下3条命令即可。

start-dfs.sh

start-yarn.sh

mr-jobhistory-daemon.sh start historyserver

start-dfs.sh命令会在节点上启动NameNode、DataNode和SecondaryNameNode服务。start-yarn.sh命令会在节点上启动ResourceManager、NodeManager服务。mr-jobhistory-daemon.sh命令会在节点上启动JobHistoryServer服务。请注意，即使对应的守护进程没有启动成功，Hadoop也不会在控制台显示错误消息，读者可以利用jps命令一步一步查询，逐步核实对应的进程是否启动成功。

2. 启动Hive CLI

启动Hive CLI测试Hive是否部署成功，使用Hive Shell的统一入口命令“hive”进入，并使用“show databases”等命令测试。依次使用的命令及执行结果如图6-41所示。

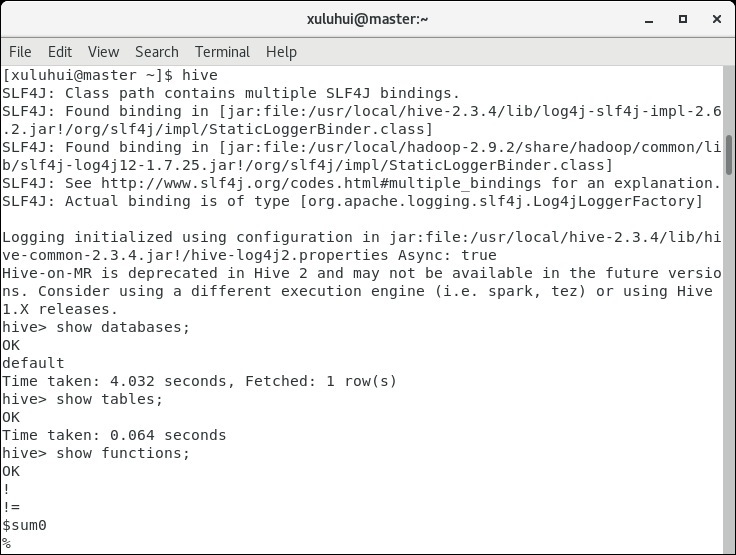


图6-41 Hive Shell统一入口命令hive

读者可以观察到，当Hive CLI启动时，在master节点上会看到多一个进程“RunJar”，若启动两个Hive CLI，会多出2个进程“RunJar”，效果如图6-42所示。

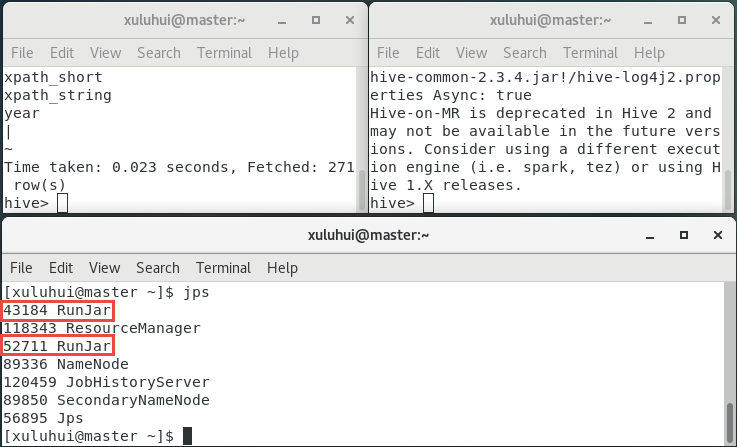


图6-42 master节点多两个进程“RunJar”

另外，读者也可以查看HDFS文件，可以看到在目录/tmp下生成了目录hive，且该目录权限为733，如图6-43所示。此时，还没有自动生成HDFS目录/user/hive/warehouse。

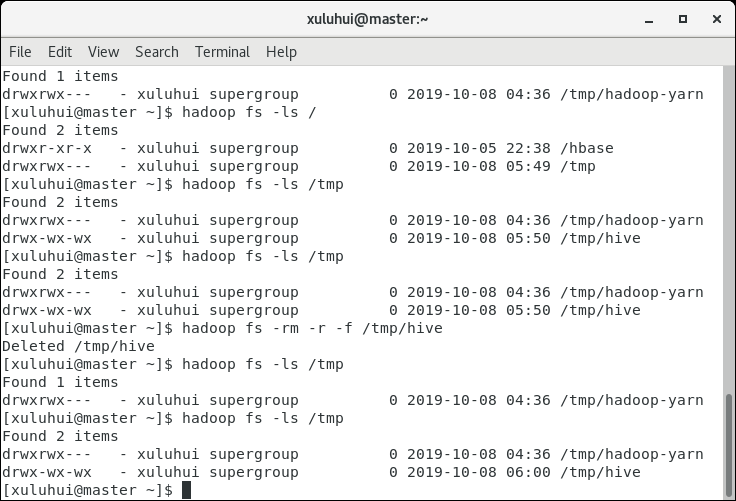


图6-43 启动Hive CLI后HDFS上的文件效果

（四）使用Hive Shell

【案例6-1】使用Hive Shell完成以下操作：

（1）进入Hive命令行接口。

（2）在Hive默认数据库default下新建student表，并将表6-15中的数据载入Hive里的student表中。

（3）编写HiveQL SELECT语句，完成以下查询：查询student表中所有记录，查询student表中所有女生记录，统计student中男女生人数。

表6-15 Hive表student的数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 性别 | 年龄 | 院系 |
| 190809011001 | xuluhui | female | 18 | bigdata |
| 190809011002 | zhouxiangzhen | female | 19 | bigdata |
| 190809011003 | liyuejun | female | 18 | bigdata |
| 190809011004 | zhangsan | male | 19 | bigdata |
| 190809101001 | lisi | male | 20 | AI |
| 190809101002 | wangwu | female | 18 | AI |

（1）使用命令“hive”进入Hive命令行，如图6-44所示。

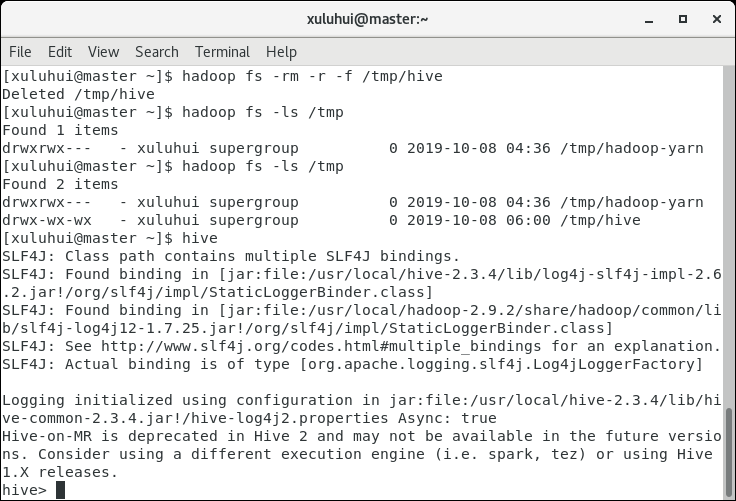


图6-44 进入Hive命令行

（2）使用“create table”命令在Hive默认数据库中创建表student。使用的HiveQL命令及结果如图6-45所示。

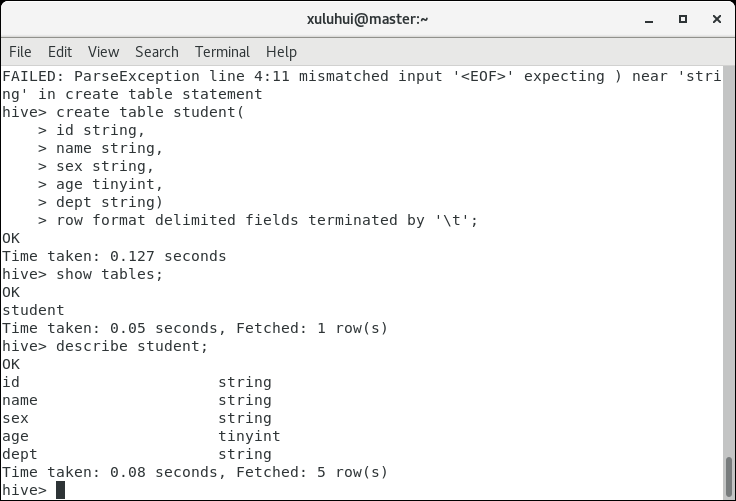


图6-45 创建Hive表student

然后，准备数据，输入数据时中间用tab键相隔。例如，在/usr/local/hive-2.3.4/testData目录下新建文件hiveStudentData.txt，以存放表中学生数据。使用的命令及结果如图7-37所示。

mkdir /usr/local/hive-2.3.4/testData

vim /usr/local/hive-2.3.4/testData/hiveStudentData.txt

在hiveStudentData.txt中手工输入以下学生数据，请注意，各数据间用“\t”相隔，这是因为创建表student时使用了语句“row format delimited fields terminated by '\t';”。

190809011001 xuluhui female 18 bigdata

190809011002 zhouxiangzhen female 19 bigdata

190809011003 liyuejun female 18 bigdata

190809011004 zhangsan male 19 bigdata

190809101001 lisi male 20 AI

190809101002 wangwu female 18 AI

最后，使用“load”命令将文件/usr/local/hive-2.3.4/testData/hiveStudentData.txt中的数据导入到Hive表student。使用的HiveQL命令及结果如图6-46所示。

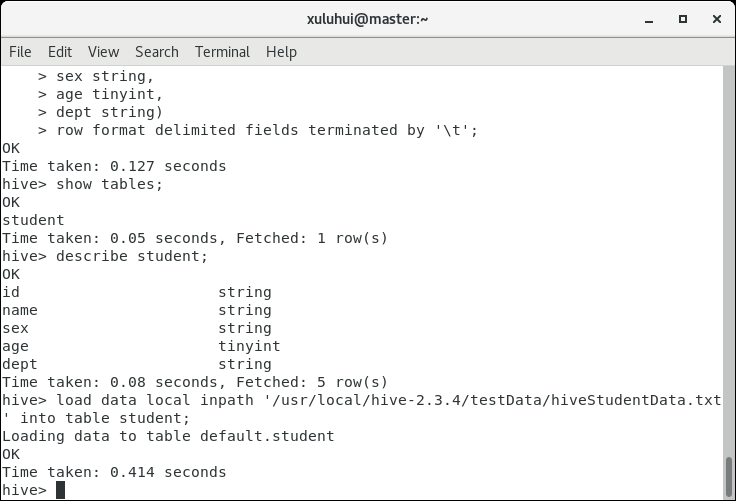


图6-46 使用“load”命令导入数据到Hive表student

（3）首先，查询member表中所有记录。使用的HiveQL命令及结果如图6-47所示。

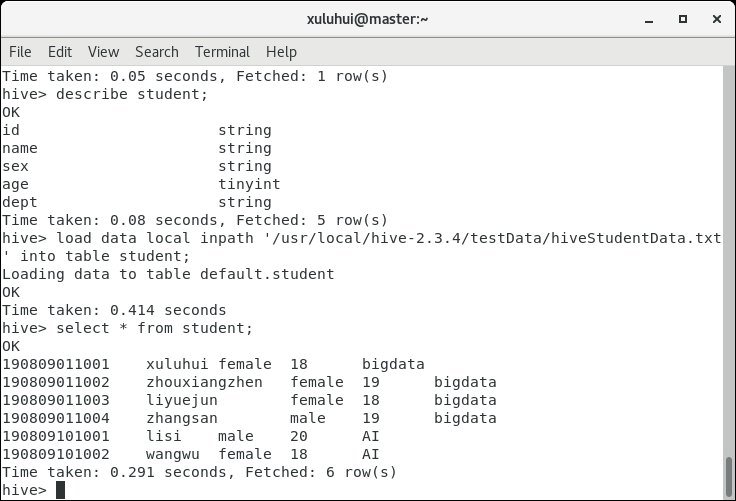


图6-47 查询student表中所有记录

其次，查询student表中所有女生记录。使用的HiveQL命令及结果如图6-48所示。

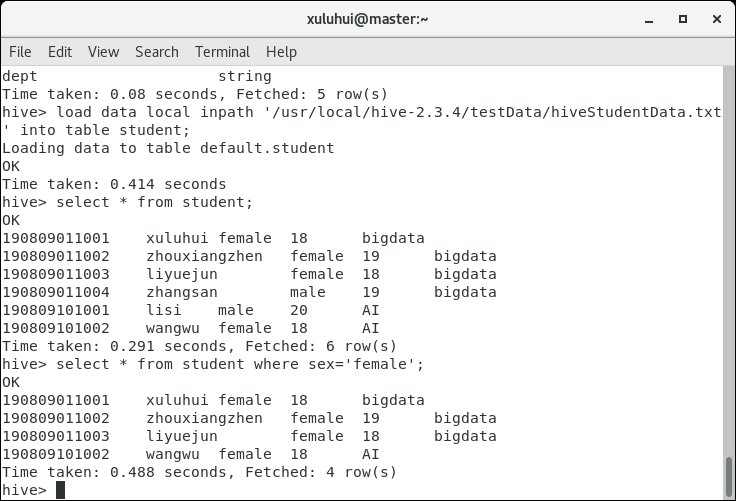


图6-48 查询student表中所有女生记录

最后，统计student中男女生人数。使用的HiveQL命令及结果如图6-49所示，可以看到HiveQL已转换为MapReduce操作。

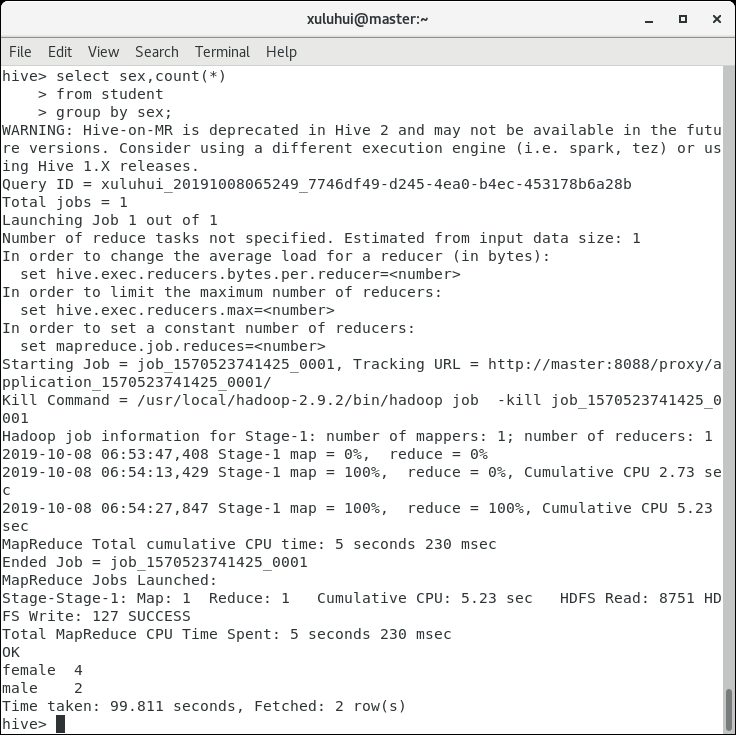


图6-49 统计student中男女生人数

实际上，创建表和导入数据到Hive表student后，即会递归生成HDFS目录/user/hive/warehouse/student，如图6-50所示。这是因为hive-default.xml配置文件中参数hive.metastore.warehouse.dir默认值为“/user/hive/warehouse”，我们在hive-site.xml文件中并未修改。

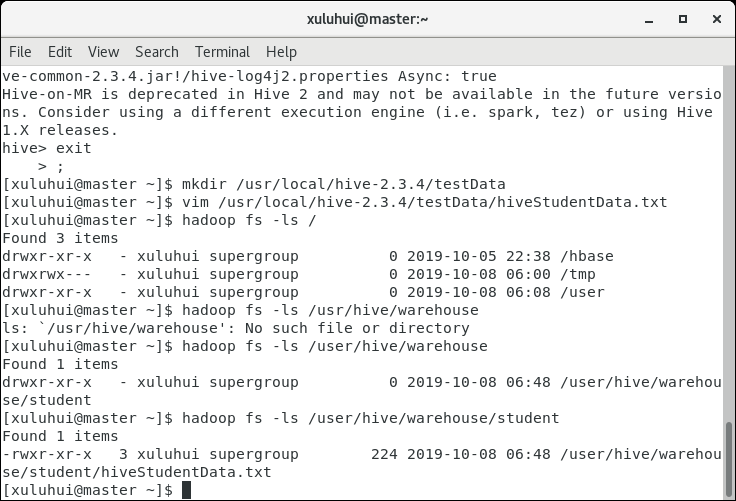


图6-50 创建表和导入数据后HDFS文件变化

此时，我们使用hive用户进入MySQL，hive数据库下拥有57个表，这些表用于存放Hive元数据，如图6-51所示。

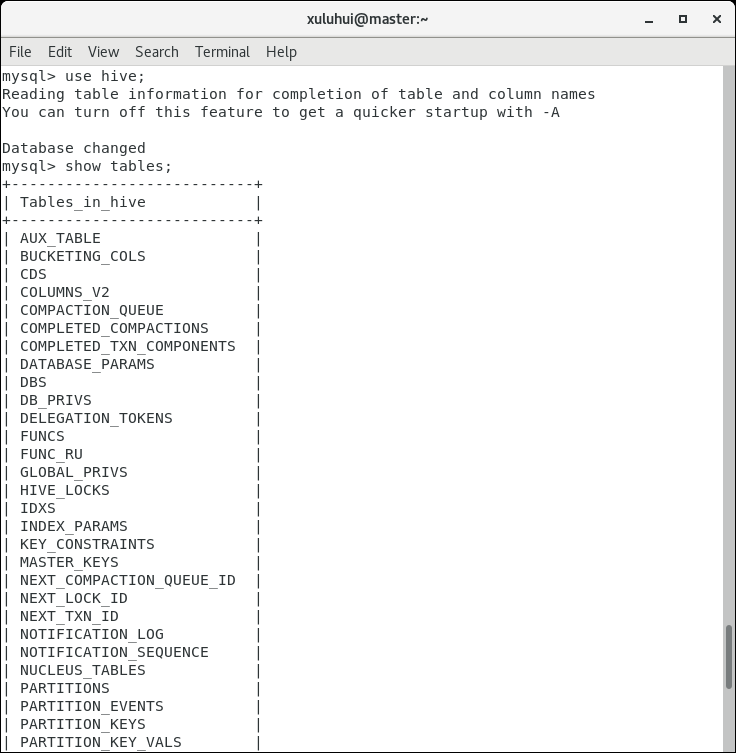


图6-51 MySQL中存放Hive Metastore的表

（4）使用命令“quit;”退出Hive CLI。

六、实验报告要求

实验报告以电子版和打印版双重形式提交。

实验报告主要内容包括实验名称、实验类型、实验地点、学时、实验环境、实验原理、实验步骤、实验结果、总结与思考等。实验报告格式如表1-9所示。

七、拓展训练

本节中，编者将通过一些实例来介绍如何使用Hive API编写应用程序，若要深入学习Hive编程，读者可以访问Hive官方网站提供的完整Hive API文档。为了提高程序编写和调试效率，本书采用Eclipse工具编写Java程序，采用版本为适用于64位Linux操作系统的Eclipse IDE 2018-09 for Java Developers。

（一）搭建HBase开发环境Eclipse

在HBase集群的主节点上搭建HBase开发环境Eclipse，具体过程请读者参考实验项目2，此处不再赘述。

（二）Hive编程实践：操纵Hive数据库和表

Java想要访问Hive，需要通过beeline的方式连接Hive，Hive Server 2提供了一个新的命令行工具beeline，Hive Server 2对之前的Hive Server做了升级，功能更加强大，它增加了权限控制，要使用beeline需要先启动Hive Server 2，再使用beeline连接。

【案例6-2】通过Hive Java API操作Hive，实现加载驱动、创建连接、创建数据库、查询所有数据库、创建表、查询所有表、查看表结构、加载数据、查询数据、统计查询（会运行MapReduce作业）、删除数据库表、删除数据库、释放资源等一系列功能。

1. 启动Hive Server 2

通过命令“hive --service hiveserver2”启动Hive Server 2，执行效果如图6-52所示。

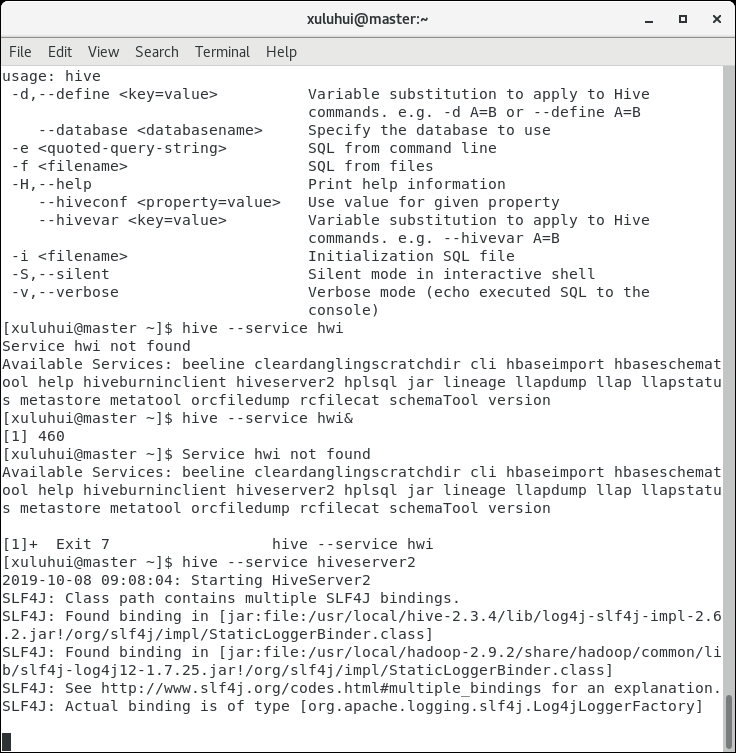


图6-52 启动Hive Server 2

2. 使用beeline连接Hive

使用如下命令通过beeline连接Hive，执行效果如图6-53所示。

beeline -u jdbc:hive2://192.168.18.130:10000/hive -n xuluhui -p

关于上述命令各个参数说明如下；

* -u：连接url，可以使用IP，也可以使用主机名，端口默认为10000。
* -n：连接的用户名，请注意，它不是登录Hive的用户名，是Hive所在服务器的登录用户名。
* -p：密码。

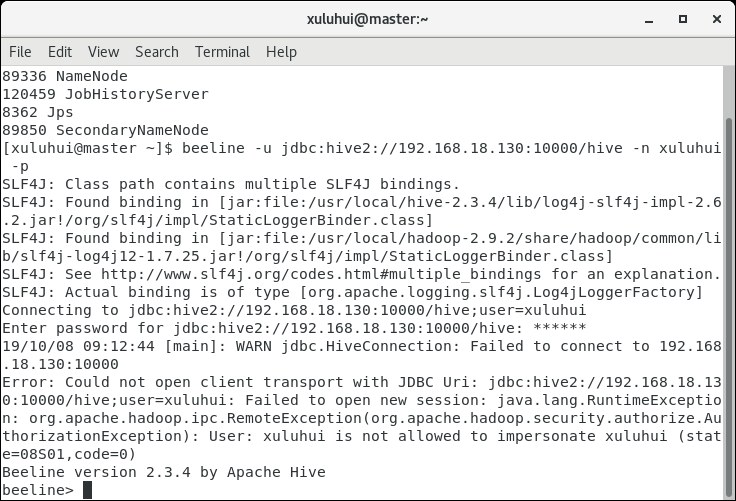


图6-53 使用beeline连接Hive

如图6-53所示，使用beeline连接时出现如下错误：

User: xuluhui is not allowed to impersonate xuluhui (state=08S01,code=0)

这是因为Hive Server 2增加了权限控制，需要在Hadoop的配置文件core-site.xml中添加如下内容，然后重启Hadoop，再使用beeline连接即可。

<property>

<name>hadoop.proxyuser.hadoop.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hadoop.groups</name>

<value>\*</value>

</property>

连接成功后，可执行与Hive Shell命令相同的命令，如果要退出连接可以使用“!q”或“!quit”命令。

若读者不熟悉beeline怎么使用，可以使用命令“beeline --help”来查看beeline的使用帮助信息，帮助信息如图所示，由于过多，只显示部分帮助，读者可部署Hive后自行查看。

Usage: java org.apache.hive.cli.beeline.BeeLine

-u <database url> the JDBC URL to connect to

-r reconnect to last saved connect url (in conjunction with !save)

-n <username> the username to connect as

-p <password> the password to connect as

-d <driver class> the driver class to use

-i <init file> script file for initialization

-e <query> query that should be executed

-f <exec file> script file that should be executed

-w (or) --password-file <password file> the password file to read password from

--hiveconf property=value Use value for given property

--hivevar name=value hive variable name and value

This is Hive specific settings in which variables

can be set at session level and referenced in Hive

commands or queries.

--property-file=<property-file> the file to read connection properties (url, driver, user, password) from

--color=[true/false] control whether color is used for display

--showHeader=[true/false] show column names in query results

--headerInterval=ROWS; the interval between which heades are displayed

--fastConnect=[true/false] skip building table/column list for tab-completion

--autoCommit=[true/false] enable/disable automatic transaction commit

--verbose=[true/false] show verbose error messages and debug info

--showWarnings=[true/false] display connection warnings

--showDbInPrompt=[true/false] display the current database name in the prompt

--showNestedErrs=[true/false] display nested errors

--numberFormat=[pattern] format numbers using DecimalFormat pattern

--force=[true/false] continue running script even after errors

--maxWidth=MAXWIDTH the maximum width of the terminal

--maxColumnWidth=MAXCOLWIDTH the maximum width to use when displaying columns

--silent=[true/false] be more silent

--autosave=[true/false] automatically save preferences

--outputformat=[table/vertical/csv2/tsv2/dsv/csv/tsv] format mode for result display

Note that csv, and tsv are deprecated - use csv2, tsv2 instead

--incremental=[true/false] Defaults to false. When set to false, the entire result set

is fetched and buffered before being displayed, yielding optimal

display column sizing. When set to true, result rows are displayed

immediately as they are fetched, yielding lower latency and

memory usage at the price of extra display column padding.

Setting --incremental=true is recommended if you encounter an OutOfMemory

on the client side (due to the fetched result set size being large).

Only applicable if --outputformat=table.

--incrementalBufferRows=NUMROWS the number of rows to buffer when printing rows on stdout,

defaults to 1000; only applicable if --incremental=true

and --outputformat=table

--truncateTable=[true/false] truncate table column when it exceeds length

--delimiterForDSV=DELIMITER specify the delimiter for delimiter-separated values output format (default: |)

--isolation=LEVEL set the transaction isolation level

--nullemptystring=[true/false] set to true to get historic behavior of printing null as empty string

--maxHistoryRows=MAXHISTORYROWS The maximum number of rows to store beeline history.

--help display this message

3. 搭建Hive开发环境Eclipse

若master节点上搭建开发环境Eclipse。若机器已安装Eclipse IDE，此小节可以忽略。

（1）获取Eclipse

Eclipse官方下载地址为https://www.eclipse.org/downloads/packages，建议读者下载较新版本，本实验选用的是2018年9月发布的Linux 64位版本Eclipse IDE 2018-09 for Java Developers，其安装包文件eclipse-java-2018-09-linux-gtk-x86\_64.tar.gz例如存放在master机器的/home/xuluhui/Downloads中。

（2）安装Eclipse

在master机器上解压eclipse-java-2018-09-linux-gtk-x86\_64.tar.gz到安装目录如/usr/local下，使用命令如下所示。

su root

cd /usr/local

tar -zxvf /home/xuluhui/Downloads/eclipse-java-2018-09-linux-gtk-x86\_64.tar.gz

4. 创建maven项目

在Eclipse中创建一个maven项目，pom.xml文件配置如下。

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.xijing</groupId>

<artifactId>hive</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.apache.hive</groupId>

<artifactId>hive-jdbc</artifactId>

<version>2.3.4</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

5. 编写Hive Java API程序

创建Java类HiveJDBC，完整代码如下所示。

package com.xijing.hive;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import java.sql.\*;

/\*\*

\* JDBC 操作 Hive（注：JDBC 访问 Hive 前需要先启动HiveServer2）

\*/

public class HiveJDBC {

private static String driverName = "org.apache.hive.jdbc.HiveDriver";

private static String url = "jdbc:hive2://192.168.18.130:10000/hive";

private static String user = "xuluhui";

private static String password = "";

private static Connection conn = null;

private static Statement stmt = null;

private static ResultSet rs = null;

// 加载驱动、创建连接

@Before

public void init() throws Exception {

Class.forName(driverName);

conn = DriverManager.getConnection(url,user,password);

stmt = conn.createStatement();

}

// 创建数据库

@Test

public void createDatabase() throws Exception {

String sql = "create database hive\_jdbc\_test";

System.out.println("Running: " + sql);

stmt.execute(sql);

}

// 查询所有数据库

@Test

public void showDatabases() throws Exception {

String sql = "show databases";

System.out.println("Running: " + sql);

rs = stmt.executeQuery(sql);

while (rs.next()) {

System.out.println(rs.getString(1));

}

}

// 创建表

@Test

public void createTable() throws Exception {

String sql = "create table student(\n" +

"id string,\n" +

"name string,\n" +

"sex string,\n" +

"age tinyint,\n" +

"dept string,\n" +

")\n" +

"row format delimited fields terminated by '\\t'";

System.out.println("Running: " + sql);

stmt.execute(sql);

}

// 查询所有表

@Test

public void showTables() throws Exception {

String sql = "show tables";

System.out.println("Running: " + sql);

rs = stmt.executeQuery(sql);

while (rs.next()) {

System.out.println(rs.getString(1));

}

}

// 查看表结构

@Test

public void descTable() throws Exception {

String sql = "desc student";

System.out.println("Running: " + sql);

rs = stmt.executeQuery(sql);

while (rs.next()) {

System.out.println(rs.getString(1) + "\t" + rs.getString(2));

}

}

// 加载数据

@Test

public void loadData() throws Exception {

String filePath = "/usr/local/hive-2.3.4/testData/hiveStudentData.txt ";

String sql = "load data local inpath '" + filePath + "' overwrite into table emp";

System.out.println("Running: " + sql);

stmt.execute(sql);

}

// 查询数据

@Test

public void selectData() throws Exception {

String sql = "select \* from emp";

System.out.println("Running: " + sql);

rs = stmt.executeQuery(sql);

System.out.println("学号" + "\t" + "姓名" + "\t" + "性别");

while (rs.next()) {

System.out.println(rs.getString("id") + "\t\t" + rs.getString("name") + "\t\t" + rs.getString("sex"));

}

}

// 统计查询（会运行MapReduce作业）

@Test

public void countData() throws Exception {

String sql = "select count(1) from student";

System.out.println("Running: " + sql);

rs = stmt.executeQuery(sql);

while (rs.next()) {

System.out.println(rs.getInt(1) );

}

}

// 删除数据库

@Test

public void dropDatabase() throws Exception {

String sql = "drop database if exists hive\_jdbc\_test";

System.out.println("Running: " + sql);

stmt.execute(sql);

}

// 删除数据库表

@Test

public void deopTable() throws Exception {

String sql = "drop table if exists student";

System.out.println("Running: " + sql);

stmt.execute(sql);

}

// 释放资源

@After

public void destory() throws Exception {

if ( rs != null) {

rs.close();

}

if (stmt != null) {

stmt.close();

}

if (conn != null) {

conn.close();

}

}

}

思考与练习题

1. 配置和使用HWI（Hive Web Interface）。

2. 在本实验案例的基础上创建外部表、分区表、分桶表，体会它们之间的区别。

参考文献

[1] CAPRIOLO E, WAMPLER D, RUTHERGLEN J. Hive编程指南[M]. 曹坤,译.北京:人民邮电出版社,2013.

[2] Apache Hive[EB/OL]. https://hive.apache.org/

[3] GitHub-Apache Hive[EB/OL]. https://github.com/apache/hive.

[4] Apache Software Foundation. Apache Hive Download[EB/OL]. https://hive.apache.org/downloads.html

[5] Apache Software Foundation. Apache Hive WIKI Confluence[EB/OL]. [2018-12-17]. https://cwiki.apache.org/confluence/display/Hive/

[6] Apache Software Foundation. Apache Hive API[EB/OL]. https://hive.apache.org/javadocs/.