Hive高级用法

实验目的

熟悉hive QL语句实现表的各种连接方式，使用hiveserver2建立hive远程连接。

实验内容

一、表连接

1.表连接（内连接，左外连接，右外连接和全外连接）

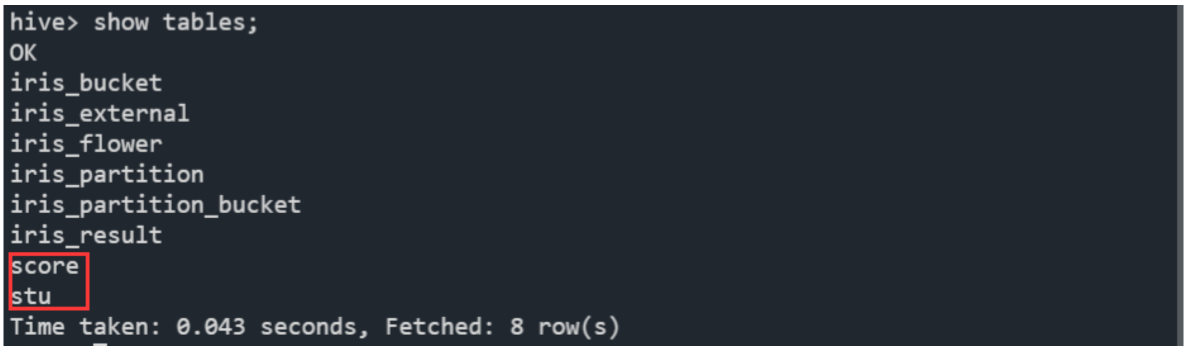
- hive只支持等值连接，即ON子句中使用等号连接，不支持非等值连接。
- 如果连接语句中有WHERE子句，会先执行JOIN子句，再执行WHERE子句。
- 可以JOIN多个表

启动hadoop并进入hive

```
start-all.sh
hive
```

建立连个表stu和score,分别从本地文件载入数据。

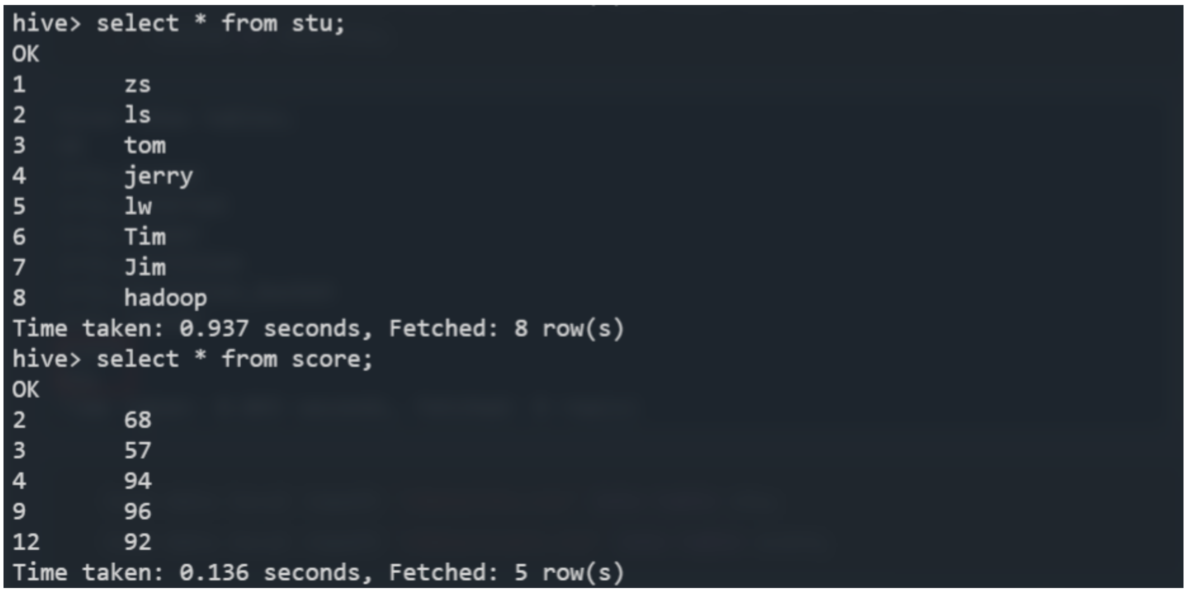
```
create table stu(id int, name string) row format delimited fields terminated by
',' stored as textfile;
create table score(s_id int, score int) row format delimited fields terminated
by ',' stored as textfile;
```



```
hive> show tables;
OK
iris_bucket
iris_external
iris_flower
iris_partition
iris_partition_bucket
iris_result
score
stu
Time taken: 0.043 seconds, Fetched: 8 row(s)
```

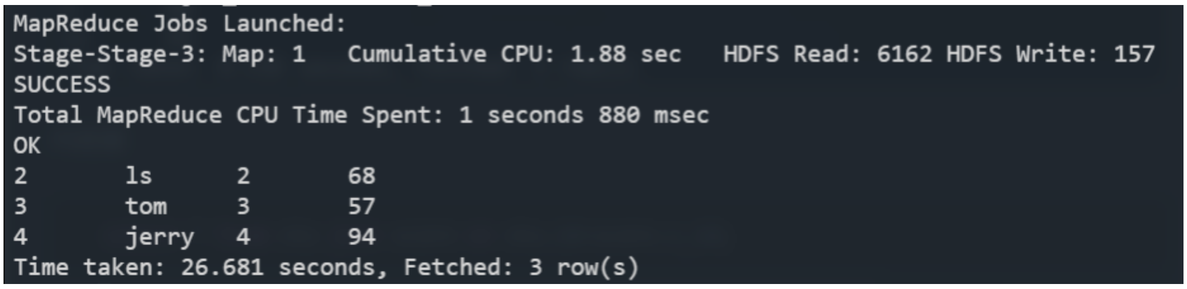
```
load data local inpath '/data/stu.csv' into table stu;
load data local inpath '/data/score.csv' into table score;
```

表中数据


```
hive> select * from stu;
OK
1       zs
2       ls
3       tom
4       jerry
5       lw
6       Tim
7       Jim
8       hadoop
Time taken: 0.937 seconds, Fetched: 8 row(s)
hive> select * from score;
OK
2       68
3       57
4       94
9       96
12      92
Time taken: 0.136 seconds, Fetched: 5 row(s)
```

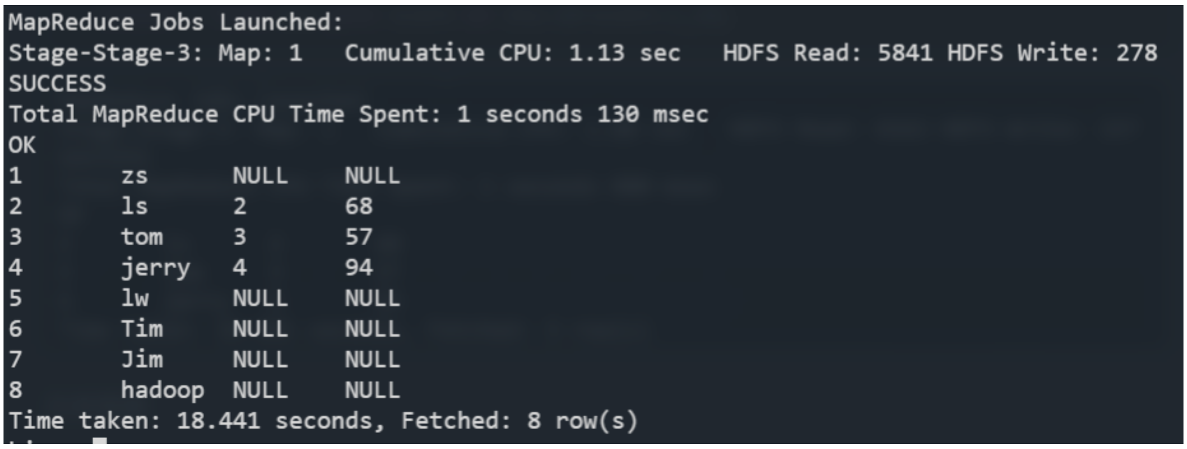
内连接（两个表的交集）

```
select * from stu join score on stu.id=score.s_id;
```


```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 1.88 sec   HDFS Read: 6162 HDFS Write: 157
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 880 msec
OK
2       ls      2       68
3       tom     3       57
4       jerry   4       94
Time taken: 26.681 seconds, Fetched: 3 row(s)
```

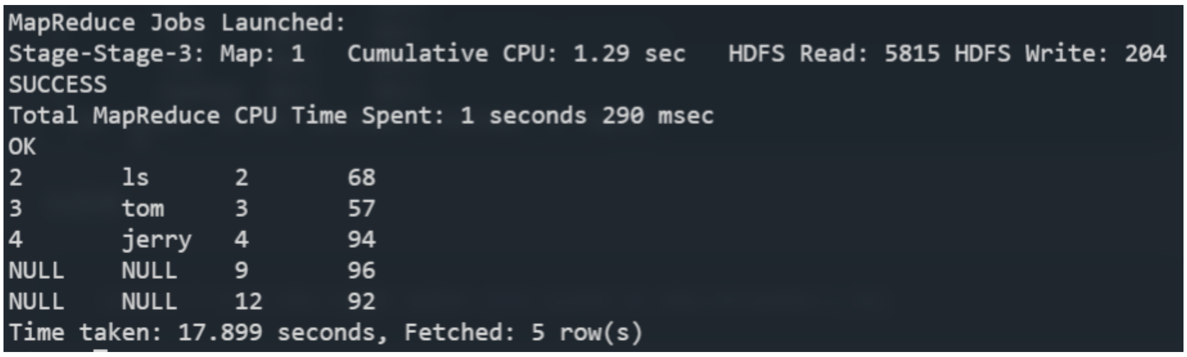
左外连接（以左表为准显示，右表没有对应的补NULL）left outer

```
select * from stu left outer join score on stu.id=score.s_id;
```


```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 1.13 sec   HDFS Read: 5841 HDFS Write: 278
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 130 msec
OK
1       zs      NULL    NULL
2       ls      2       68
3       tom     3       57
4       jerry   4       94
5       lw      NULL    NULL
6       Tim     NULL    NULL
7       Jim     NULL    NULL
8       hadoop  NULL    NULL
Time taken: 18.441 seconds, Fetched: 8 row(s)
```

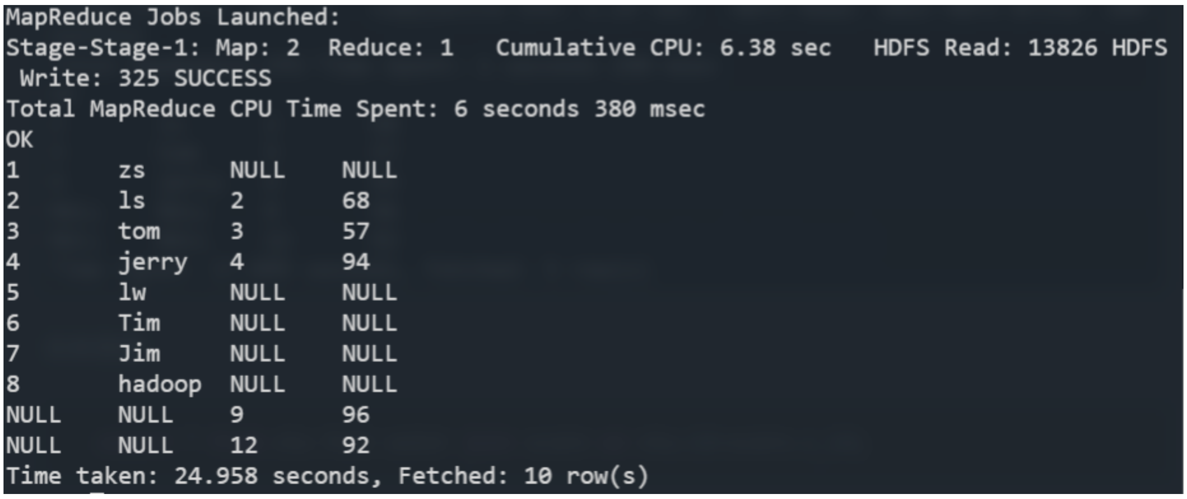
右连接（以右表为准，左表没有对应的补NULL） right outer

```
select * from stu right outer join score on stu.id=score.s_id;
```


```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 1.29 sec   HDFS Read: 5815 HDFS Write: 204
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 290 msec
OK
2       ls      2       68
3       tom     3       57
4       jerry   4       94
NULL    NULL    9       96
NULL    NULL    12      92
Time taken: 17.899 seconds, Fetched: 5 row(s)
```

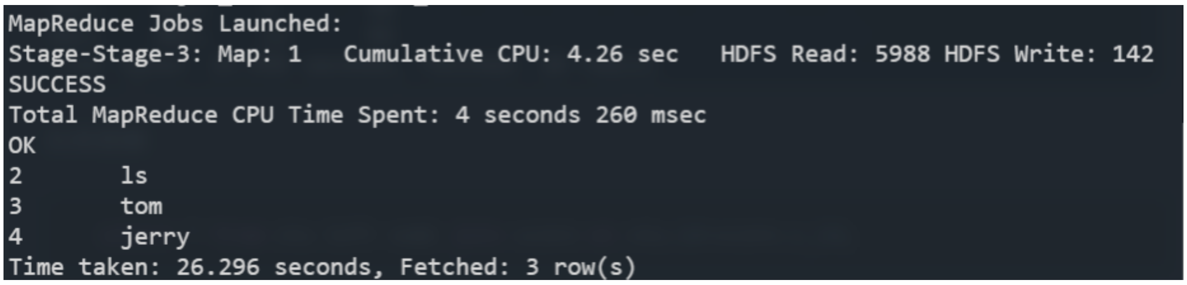
全外连接（两个表的并集）full outer

```
select * from stu full outer join score on stu.id=score.s_id;
```


```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 6.38 sec   HDFS Read: 13826 HDFS
 Write: 325 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 380 msec
OK
1       zs      NULL    NULL
2       ls      2       68
3       tom     3       57
4       jerry   4       94
5       lw      NULL    NULL
6       Tim     NULL    NULL
7       Jim     NULL    NULL
8       hadoop  NULL    NULL
NULL    NULL    9       96
NULL    NULL    12      92
Time taken: 24.958 seconds, Fetched: 10 row(s)
```

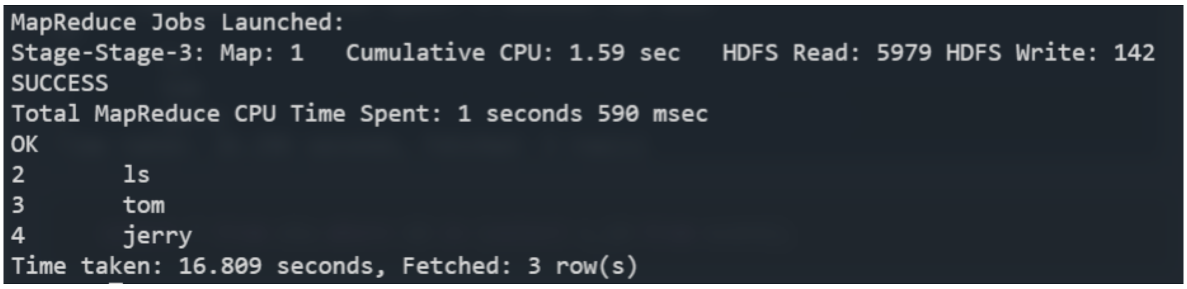
左半连接（相当于内连接后去掉右表数据）left semi

```
select * from stu left semi join score on stu.id=score.s_id;
```


```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 4.26 sec   HDFS Read: 5988 HDFS Write: 142
SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 260 msec
OK
2       ls
3       tom
4       jerry
Time taken: 26.296 seconds, Fetched: 3 row(s)
```

相当于下面 in，而且这条执行的更快。1.59s

```
select * from stu where id in (select s_id from score);
```


```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 1.59 sec   HDFS Read: 5979 HDFS Write: 142
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 590 msec
OK
2       ls
3       tom
4       jerry
Time taken: 16.809 seconds, Fetched: 3 row(s)
```

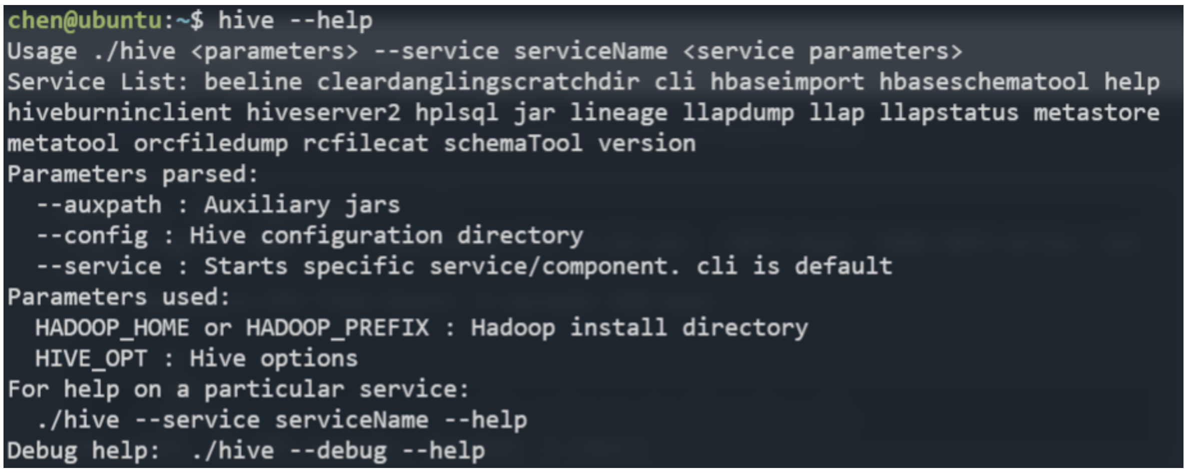
二、Hive的连接

CLI连接

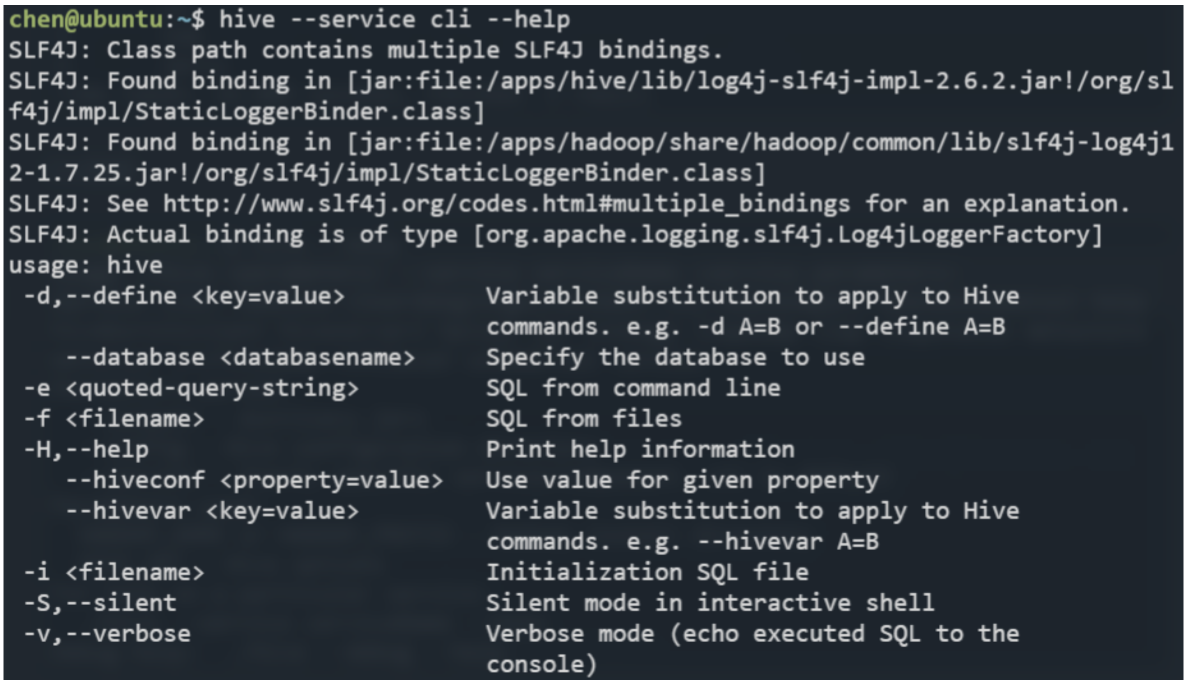
在命令行直接执行hive相当于执行hive --service cli;

使用hive --help可以看hive命令能够启动哪些服务;

通过 hive --service serviceName --help 可以查看某个具体命令的使用方式。



```
chen@ubuntu:~$ hive --help
Usage ./hive <parameters> --service serviceName <service parameters>
Service List: beeline cleardanglingscratchdir cli hbaseimport hbaseschematool help
hiveburninclient hiveserver2 hplsql jar lineage llapdump llap llapstatus metastore
metatool orcfiledump rcfilecat schemaTool version
Parameters parsed:
  --auxpath : Auxiliary jars
  --config : Hive configuration directory
  --service : Starts specific service/component. cli is default
Parameters used:
  HADOOP_HOME or HADOOP_PREFIX : Hadoop install directory
  HIVE_OPT : Hive options
For help on a particular service:
  ./hive --service serviceName --help
Debug help:  ./hive --debug --help
```



```
chen@ubuntu:~$ hive --service cli --help
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log4j1
2-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
usage: hive
 -d,--define <key=value>          Variable substitution to apply to Hive
                                  commands. e.g. -d A=B or --define A=B
    --database <databasename>     Specify the database to use
 -e <quoted-query-string>         SQL from command line
 -f <filename>                    SQL from files
 -H,--help                        Print help information
    --hiveconf <property=value>   Use value for given property
    --hivevar <key=value>         Variable substitution to apply to Hive
                                  commands. e.g. --hivevar A=B
 -i <filename>                    Initialization SQL file
 -S,--silent                      Silent mode in interactive shell
 -v,--verbose                     Verbose mode (echo executed SQL to the
                                  console)
```

HiveServer2

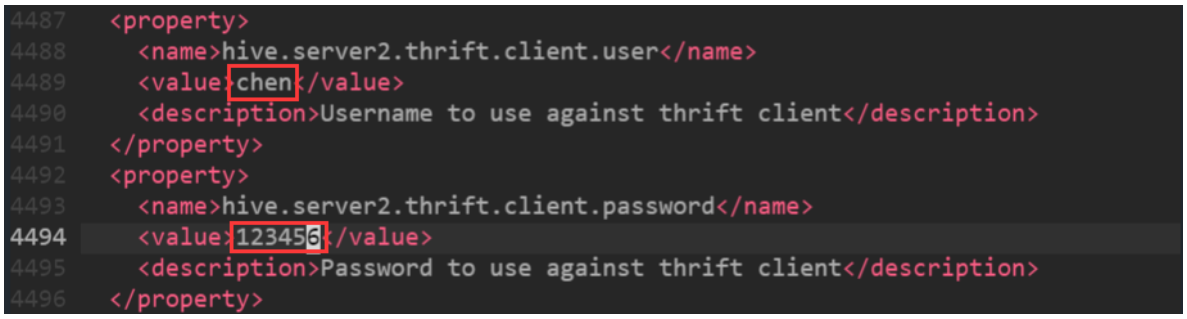
HiveServer 允许远程客户端可以使用各种编程语言向 Hive 提交请求并检索结果。 HiveServer2 是 HiveServer 改进版本，提供了新的 ThriftAPI 来处理 JDBC 或者 ODBC 客 户端，可以进行 Kerberos 身份验证，支持多个客户端并发。

Beeline

HiveServer2 提供了新的 CLI：BeeLine，它是基于 SQLLine CLI 的 JDBC 客户端。Hive CLI 与 Beeline 均为控制台命令行操作模式，区别在于 Hive CLI 只能操作本地 Hive 服务， 而 Beeline 可以通过 JDBC 连接远程服 。

Beeline 使用密码登录 Hive，需要设置用户名和密码，在 hive-site.xml 4489 行和 4494 行做如下修改，将两处的值 anonymous，改为自己的用户名和密码。

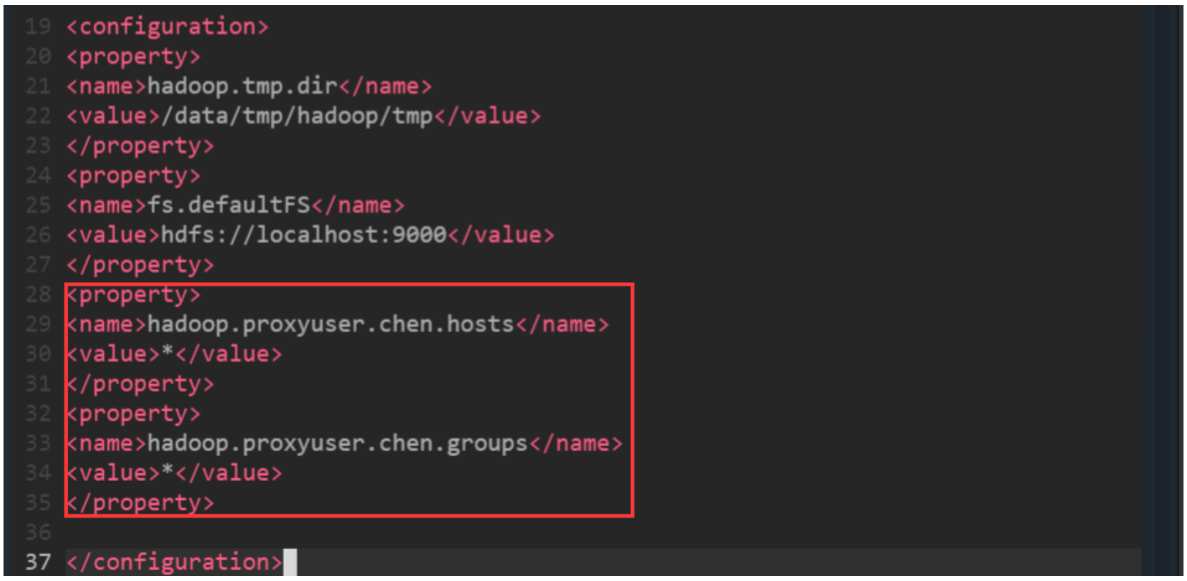
/apps/hive/conf/hive-site.xml



```
<property>
  <name>hive.server2.thrift.client.user</name>
  <value>chen</value>
  <description>Username to use against thrift client</description>
</property>
<property>
  <name>hive.server2.thrift.client.password</name>
  <value>123456</value>
  <description>Password to use against thrift client</description>
</property>
```

在/apps/hadoop/etc/hadoop/core-site.xml中添加配置

- hadoop.proxyuser.chen.hosts 表示代理用户chen的所属组在任意节点都能访问 HDFS 集群
- hadoop.proxyuser.chen.groups 表示任意节点使用 Hadoop 集群的代理用户chen都能访问 HDFS 集群



```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/data/tmp/hadoop/tmp</value>
</property>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.proxyuser.chen.hosts</name>
<value>*</value>
</property>
<property>
<name>hadoop.proxyuser.chen.groups</name>
<value>*</value>
</property>

</configuration>
```

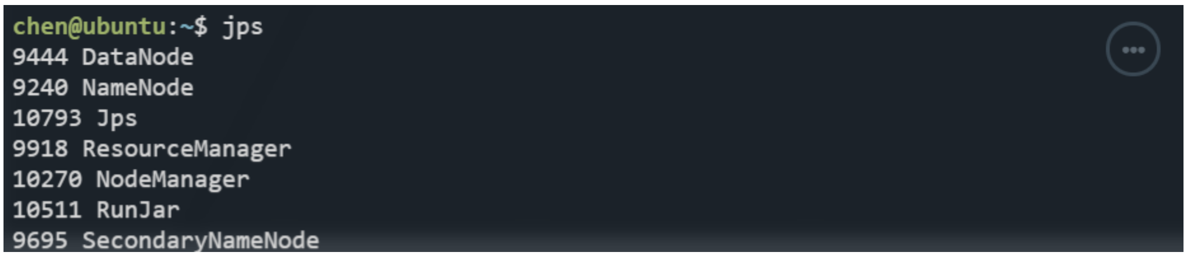
在做完以上配置后，hadoop需要重启。

```
stop-all.sh
start-all.sh
```

后台启动hiveserver2

```
hiveserver2 &
```

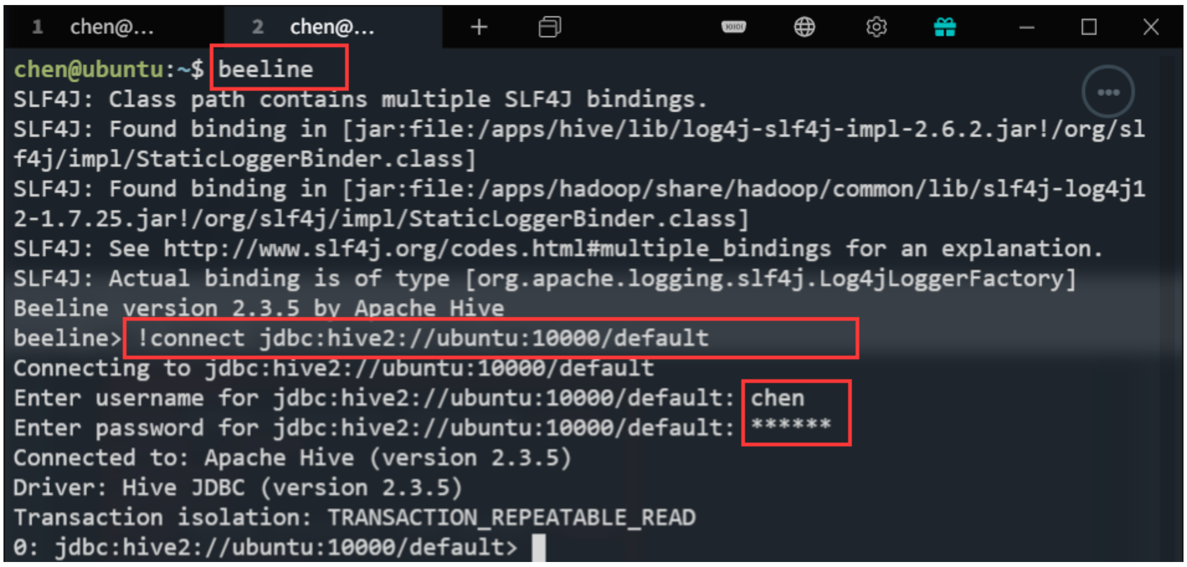
等待几秒钟后，可以看到RunJar这个进程。



```
chen@ubuntu:~$ jps
9444 DataNode
9240 NameNode
10793 Jps
9918 ResourceManager
10270 NodeManager
10511 RunJar
9695 SecondaryNameNode
```

另一个终端

第一种连接方式：



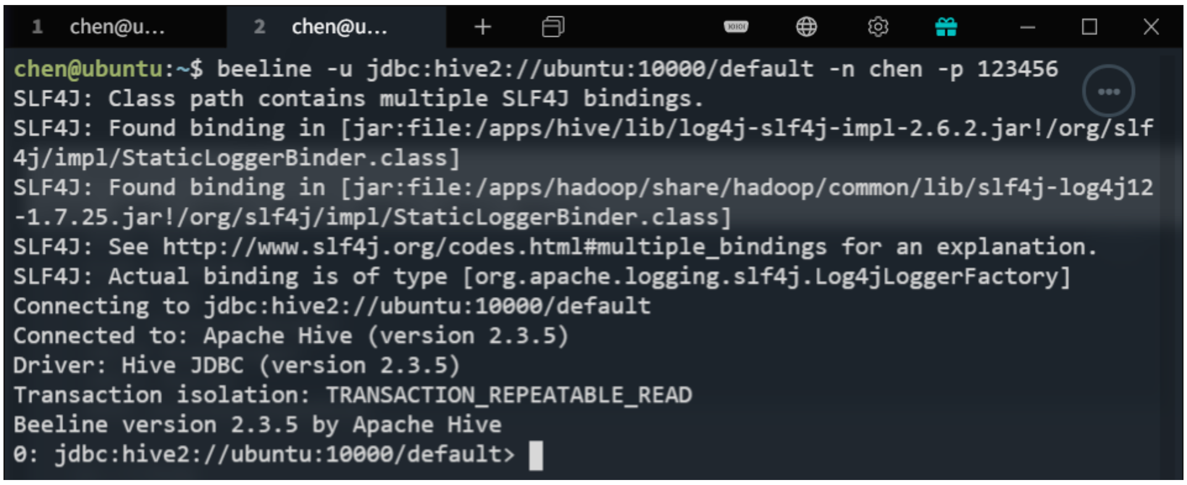
```
chen@ubuntu:~$ beeline
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log4j1
2-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 2.3.5 by Apache Hive
beeline> !connect jdbc:hive2://ubuntu:10000/default
Connecting to jdbc:hive2://ubuntu:10000/default
Enter username for jdbc:hive2://ubuntu:10000/default: chen
Enter password for jdbc:hive2://ubuntu:10000/default: ******
Connected to: Apache Hive (version 2.3.5)
Driver: Hive JDBC (version 2.3.5)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://ubuntu:10000/default>
```

第二种连接方式：

直接在命令行执行

- -u 指定元数据库的链接信息
- -n 指定用户名
- -p 指定密码

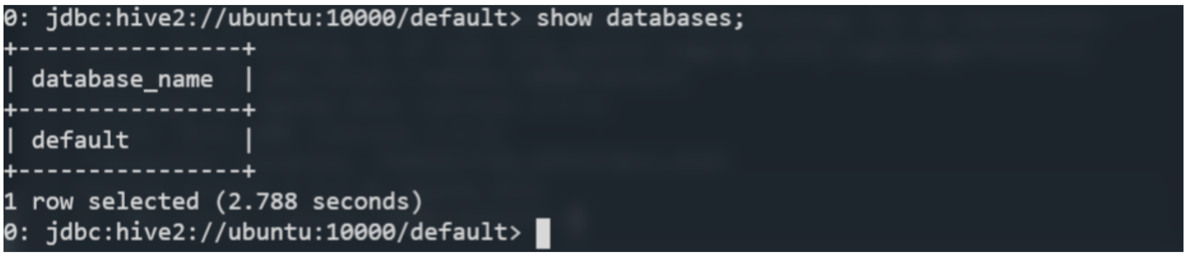
```
beeline -u jdbc:hive2://ubuntu:10000/default -n chen -p 123456
```



```
chen@ubuntu:~$ beeline -u jdbc:hive2://ubuntu:10000/default -n chen -p 123456
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log4j12
-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://ubuntu:10000/default
Connected to: Apache Hive (version 2.3.5)
Driver: Hive JDBC (version 2.3.5)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.5 by Apache Hive
0: jdbc:hive2://ubuntu:10000/default>
```

远程登录成功

```
show databases;
```



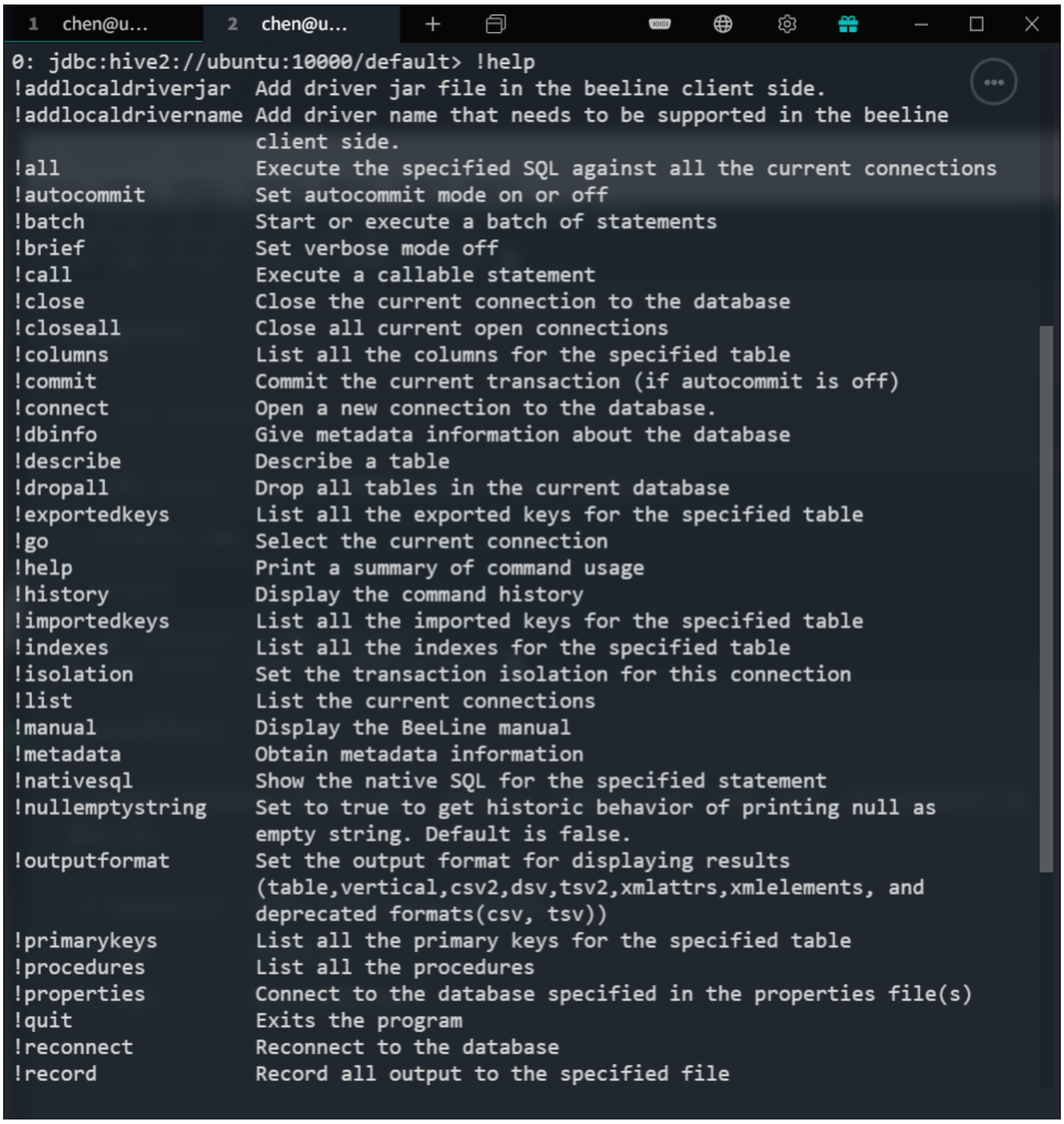
```
0: jdbc:hive2://ubuntu:10000/default> show databases;
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
1 row selected (2.788 seconds)
0: jdbc:hive2://ubuntu:10000/default>
```

Beeline管理命令

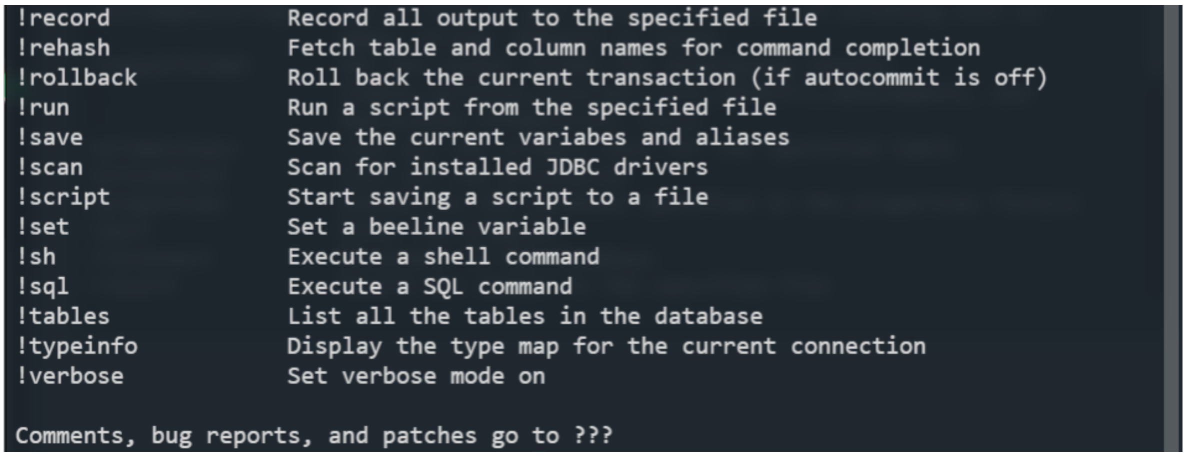
beeline执行查询都是正常的SQL输入，管理命令，比如进行连接，中断，退出，命令前需要加!不需要终止符。

- !connect url　连接不同的hiveserver2服务器

- !exit　　　退出shell
- !help　　　显示全部管理命令

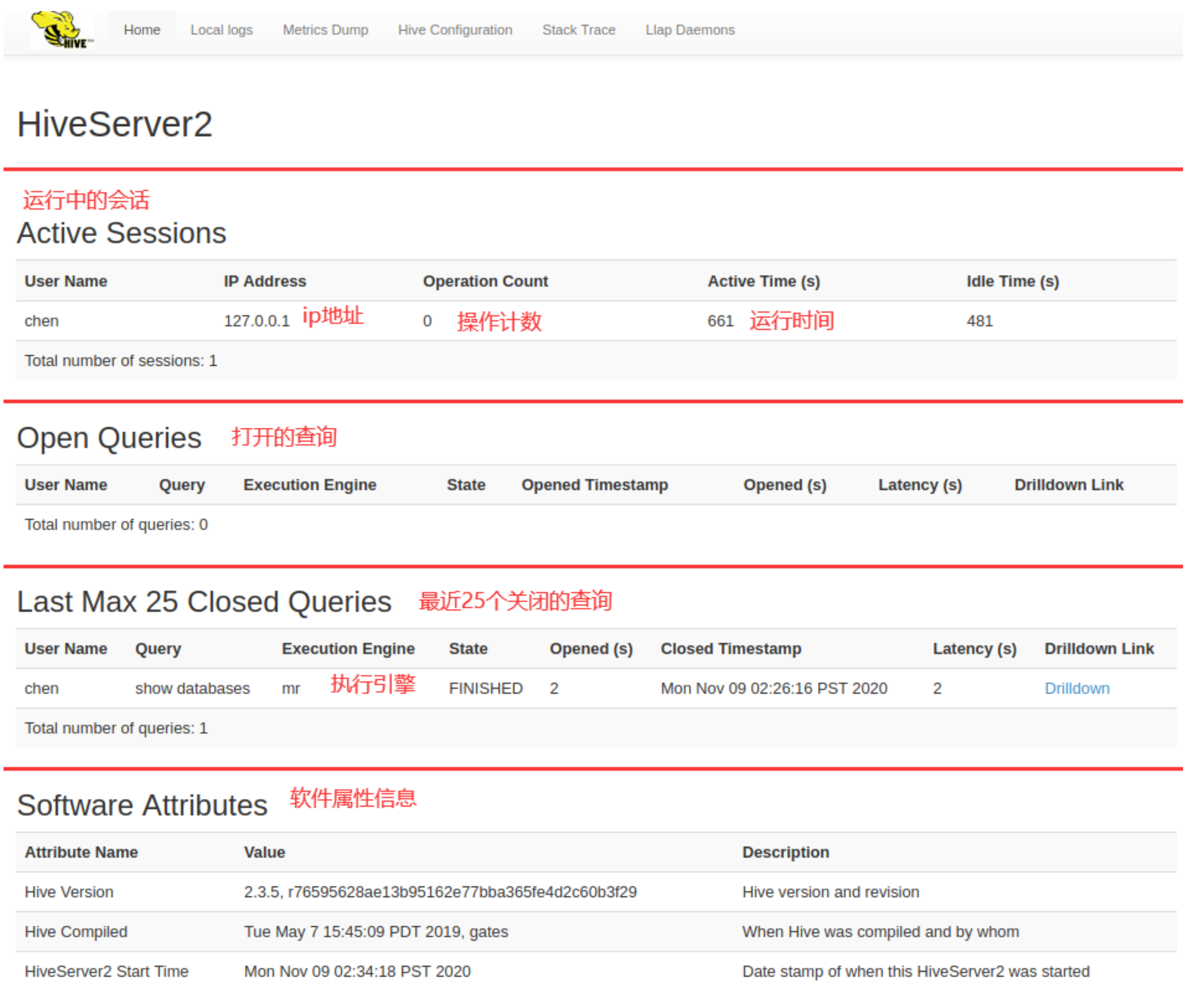


```
0: jdbc:hive2://ubuntu:10000/default> !help
!addlocaldriverjar  Add driver jar file in the beeline client side.
!addlocaldrivername Add driver name that needs to be supported in the beeline
                    client side.
!all                Execute the specified SQL against all the current connections
!autocommit         Set autocommit mode on or off
!batch              Start or execute a batch of statements
!brief              Set verbose mode off
!call               Execute a callable statement
!close              Close the current connection to the database
!closeall           Close all current open connections
!columns            List all the columns for the specified table
!commit             Commit the current transaction (if autocommit is off)
!connect            Open a new connection to the database.
!dbinfo             Give metadata information about the database
!describe           Describe a table
!dropall            Drop all tables in the current database
!exportedkeys       List all the exported keys for the specified table
!go                 Select the current connection
!help               Print a summary of command usage
!history            Display the command history
!importedkeys       List all the imported keys for the specified table
!indexes            List all the indexes for the specified table
!isolation          Set the transaction isolation for this connection
!list               List the current connections
!manual             Display the BeeLine manual
!metadata           Obtain metadata information
!nativesql          Show the native SQL for the specified statement
!nullemptystring    Set to true to get historic behavior of printing null as
                    empty string. Default is false.
!outputformat       Set the output format for displaying results
                    (table,vertical,csv2,dsv,tsv2,xmlattrs,xmlelements, and
                    deprecated formats(csv, tsv))
!primarykeys        List all the primary keys for the specified table
!procedures         List all the procedures
!properties         Connect to the database specified in the properties file(s)
!quit               Exits the program
!reconnect          Reconnect to the database
!record             Record all output to the specified file
```



```
!record             Record all output to the specified file
!rehash             Fetch table and column names for command completion
!rollback           Roll back the current transaction (if autocommit is off)
!run                Run a script from the specified file
!save               Save the current variabes and aliases
!scan               Scan for installed JDBC drivers
!script             Start saving a script to a file
!set                Set a beeline variable
!sh                 Execute a shell command
!sql                Execute a SQL command
!tables             List all the tables in the database
!typeinfo           Display the type map for the current connection
!verbose            Set verbose mode on

Comments, bug reports, and patches go to ???
```

hiveserver2 webUI 界面



HiveServer2

Active Sessions

User Name	IP Address	Operation Count	Active Time (s)	Idle Time (s)
chen	127.0.0.1	0	661	481
Total number of sessions: 1				

Open Queries

User Name	Query	Execution Engine	State	Opened Timestamp	Opened (s)	Latency (s)	Drilldown Link
Total number of queries: 0							

Last Max 25 Closed Queries

User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
chen	show databases	mr	FINISHED	2	Mon Nov 09 02:26:16 PST 2020	2	Drilldown
Total number of queries: 1							

Software Attributes

Attribute Name	Value	Description
Hive Version	2.3.5, r76595628ae13b95162e77bba365fe4d2c60b3f29	Hive version and revision
Hive Compiled	Tue May 7 15:45:09 PDT 2019, gates	When Hive was compiled and by whom
HiveServer2 Start Time	Mon Nov 09 02:34:18 PST 2020	Date stamp of when this HiveServer2 was started

三、Hive JDBC

类似于java访问关系型数据库，主要是URL和驱动不一样，而且主要是查询数据，不可以删除和更新数据。

hive JDBC 的连接参数如下：

驱动名：org.apache.hive.jdbc.HiveDriver

连接字符串：jdbc:hive2://主机:端口/数据库名（默认数据库是default）

HiveService.java

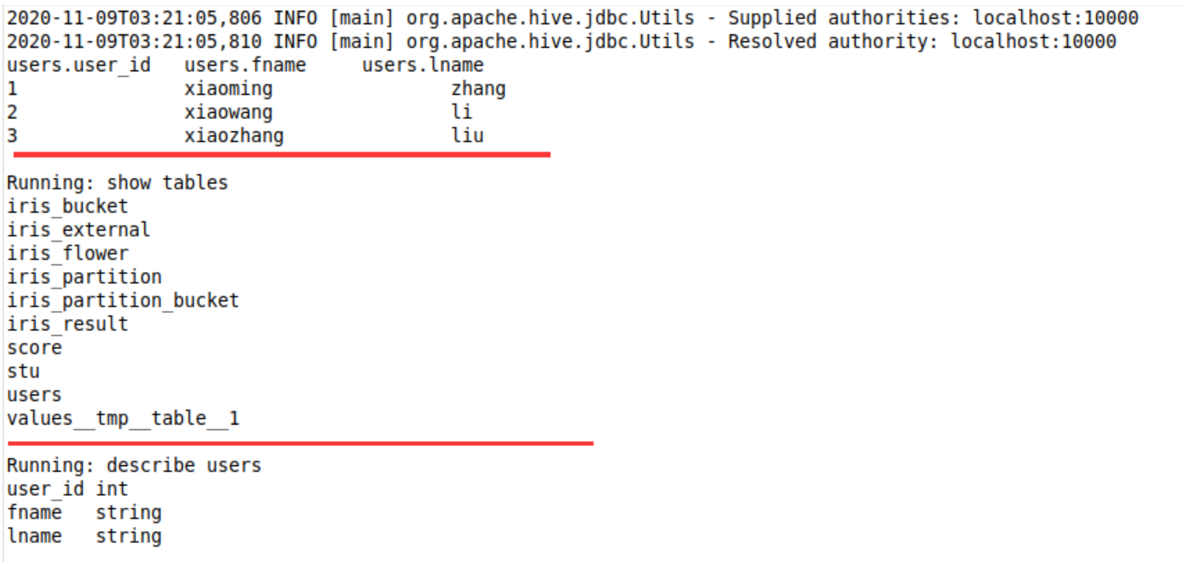
```java
package dsd.hive;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import org.apache.log4j.Logger;

public class HiveService {
    static Logger logger = Logger.getLogger(HiveService.class);
    // hive的jdbc驱动类
    public static String dirverName = "org.apache.hive.jdbc.HiveDriver";
    // 连接hive的URL hive2版本需要的是jdbc:hive2，而不是 jdbc:hive
    public static String url = "jdbc:hive2://localhost:10000";
    // 登录linux的用户名  一般会给权限大一点的用户，否则无法进行事务型操作
    public static String user = "chen";
    // 登录linux的密码
    public static String pass = "123456";

    /**
     * 创建连接
     *
     * @return
     * @throws SQLException
     */
    public static Connection getConn() {
        Connection conn = null;
        try {
            Class.forName(dirverName);
            conn = DriverManager.getConnection(url, user, pass);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return conn;
    }

    /**
     * 创建命令
     *
     * @param conn
     * @return
     * @throws SQLException
     */
    public static Statement getStmt(Connection conn) throws SQLException {
        logger.debug(conn);
        if (conn == null) {
            logger.debug("this conn is null");
        }
        return conn.createStatement();
    }

    /**
     * 关闭连接
     *
     * @param conn
     */
    public static void closeConn(Connection conn) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
     * 关闭命令
     *
     * @param stmt
     */
    public static void closeStmt(Statement stmt) {
        try {
            stmt.close();
        } catch (SQLException e) {
                e.printStackTrace();
            }
        }
}
```

HiveTest.java

```
package dsd.hive;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import org.apache.log4j.Logger;

public class HiveTest {

    static Logger logger = Logger.getLogger(HiveTest.class);

    public static void main(String[] args) {

        Connection conn = HiveService.getConn();
        Statement stmt = null;
        try {

            stmt = HiveService.getStmt(conn);
            stmt.execute("drop table if exists users");// 需要拥有hdfs文件读写权限的用户才可以进行此操作
            logger.debug("drop table is susscessful");

            stmt.execute("create table users(user_id int, fname string,lname string )  row format delimited fields terminated by ','");// 需要拥有hdfs文件读写权限的用户才可以进行此操作
            logger.debug("create table is susscessful");

            stmt.execute("insert into users(user_id, fname,lname) values(100,'hongda','chen')");// 需要拥有hdfs文件读写权限的用户才可以进行此操作
            logger.debug("insert is susscessful");

            stmt.execute("load data local inpath '/home/chen/hive_higher/user.txt' overwrite into table users");// 需要拥有hdfs文件读写权限的用户才可以进行此操作
            logger.debug("load data is susscessful");

            String sql = "select * from users";

            ResultSet res = null;
            res = stmt.executeQuery(sql);

            ResultSetMetaData meta = res.getMetaData();   //fields name

            for (int i = 1; i <= meta.getColumnCount(); i++) {
                System.out.print(meta.getColumnName(i) + "\t");
            }
            System.out.println();
            while (res.next()) {
                System.out.print(res.getInt(1) + "\t\t");
                System.out.print(res.getString(2) + "\t\t");
                System.out.print(res.getString(3));
                System.out.println();
            }

            sql = "show tables ";
            System.out.println("\nRunning: " + sql);
            res = stmt.executeQuery(sql);
            while (res.next()) {
                System.out.println(res.getString(1));
            }
            // describe table
            sql = "describe users";
            System.out.println("\nRunning: " + sql);
            res = stmt.executeQuery(sql);
            while (res.next()) {
                System.out.println(res.getString(1) + "\t" + res.getString(2));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        HiveService.closeStmt(stmt);
        HiveService.closeConn(conn);
    }
}
```

建立hive_api项目。导入/apps/hive/lib下的所有jar包。启动hiveserver2

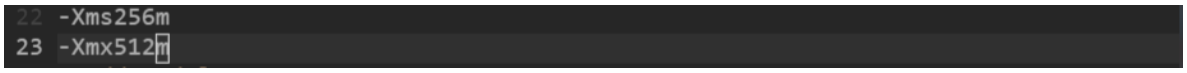
运行HiveTest.java　　run as javaApplication



```
2020-11-09T03:21:05,806 INFO [main] org.apache.hive.jdbc.Utils - Supplied authorities: localhost:10000
2020-11-09T03:21:05,810 INFO [main] org.apache.hive.jdbc.Utils - Resolved authority: localhost:10000
users.user_id   users.fname     users.lname
1               xiaoming                zhang
2               xiaowang                li
3               xiaozhang               liu

Running: show tables
iris_bucket
iris_external
iris_flower
iris_partition
iris_partition_bucket
iris_result
score
stu
users
values__tmp__table__1

Running: describe users
user_id int
fname   string
lname   string
```

eclipse闪退的问题

在/apps/eclipse/eclipse.ini修改23行为512



```
22 -Xms256m
23 -Xmx512m
```

四、Hive UDF（user-defined functions）

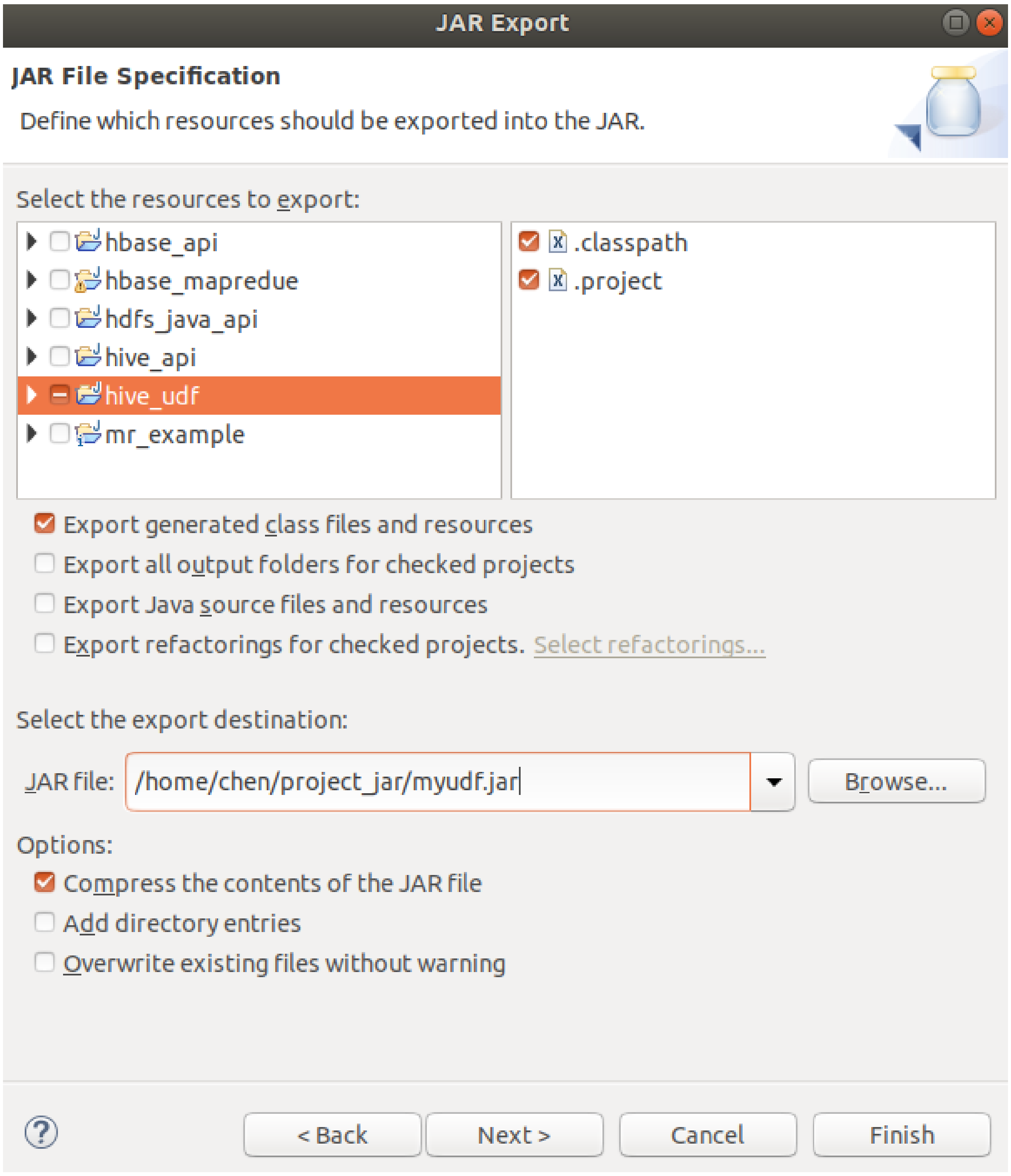
注意

- 自定义UDF需要继承org.apache.hadoop.hive.ql.exec.UDF
- 需要实现evaluate函数
- evaluate 函数支持重载

新建工程【hive_udf】并创建包【sds.hive_udf】和类【StringExt】。将 Hive 安装目录下的 lib 文件夹中的全部 Jar 文件作为外部 Jar 文件导入。

StringExt.java

```java
package sds.hive_udf;

import org.apache.hadoop.hive.ql.exec.UDF;

/**
 * 自定义Hive函数，需要继承org.apache.hadoop.hive.ql.exec.UDF 并实现evaluate方法
 *
 */
public class StringExt extends UDF {

    public String evaluate(String name) {
        return "Hello " + name;
    }

    // 添加一个空的main方法是为了使用eclipse工具打成jar包时方便；
    // 如果没有main方法，不能使用eclipse工具可视化打成jar包
    public static void main(String[] args) {

    }
}
```


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [ ] hbase_api
- [ ] hbase_mapredue
- [ ] hdfs_java_api
- [ ] hive_api
- [–] hive_udf
- [ ] mr_example

- [✓] .classpath
- [✓] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

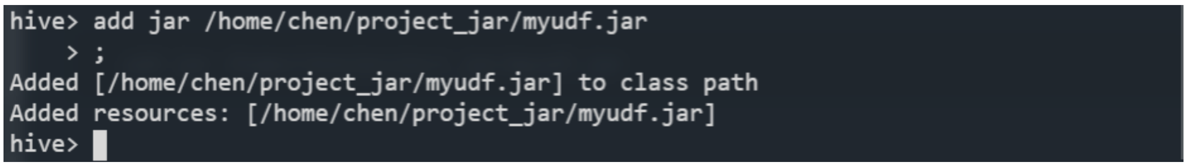
JAR file: /home/chen/project_jar/myudf.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

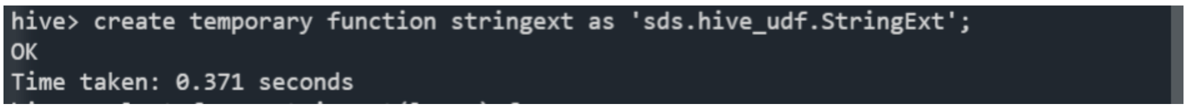
使用 add 命令将 Jar 包导入到 Hive 的 ClassPath，如果 jar 包 myudf.jar 放在 $HIVE_HOME/lib 目录下，这一步可以省略。

```
add jar /home/chen/project_jar/myudf.jar
```


```
hive> add jar /home/chen/project_jar/myudf.jar
    > ;
Added [/home/chen/project_jar/myudf.jar] to class path
Added resources: [/home/chen/project_jar/myudf.jar]
hive> 
```

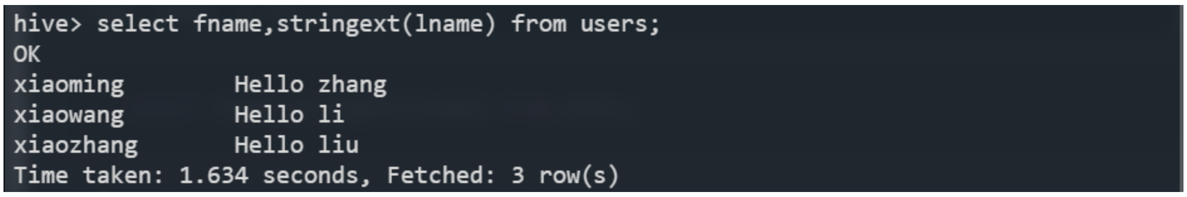
创建临时函数

```
create temporary function stringext as 'sds.hive_udf.StringExt';
```


```
hive> create temporary function stringext as 'sds.hive_udf.StringExt';
OK
Time taken: 0.371 seconds
```

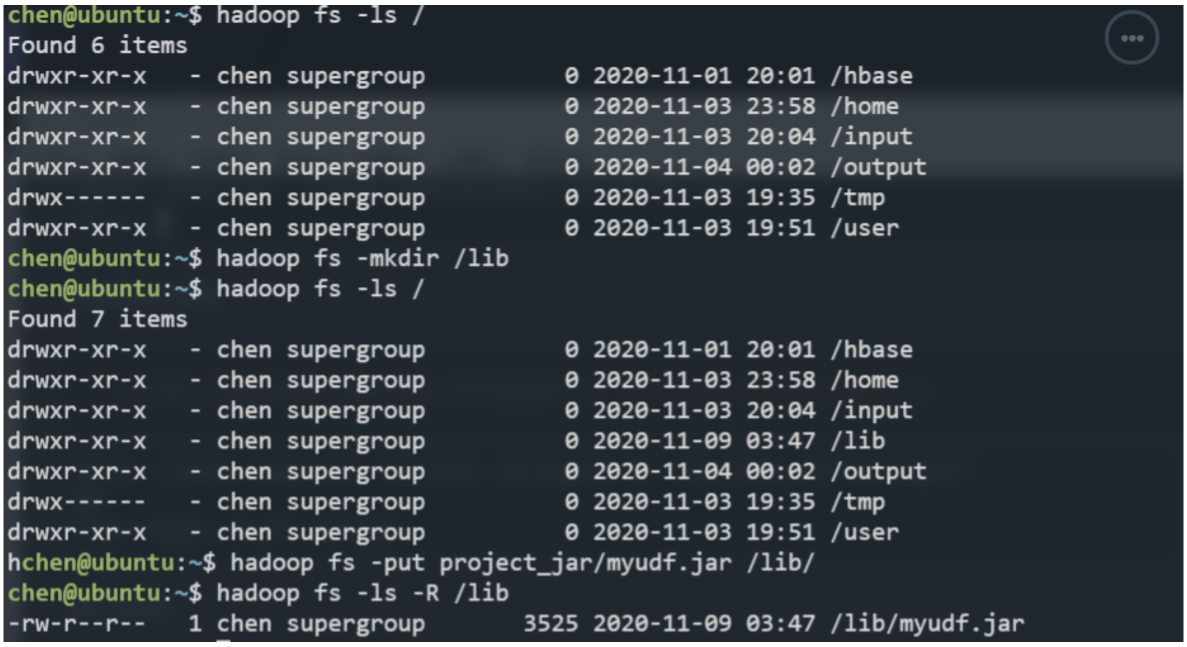
执行结果

```
select fname,stringext(lname) from users;
```



```
hive> select fname,stringext(lname) from users;
OK
xiaoming        Hello zhang
xiaowang        Hello li
xiaozhang       Hello liu
Time taken: 1.634 seconds, Fetched: 3 row(s)
```

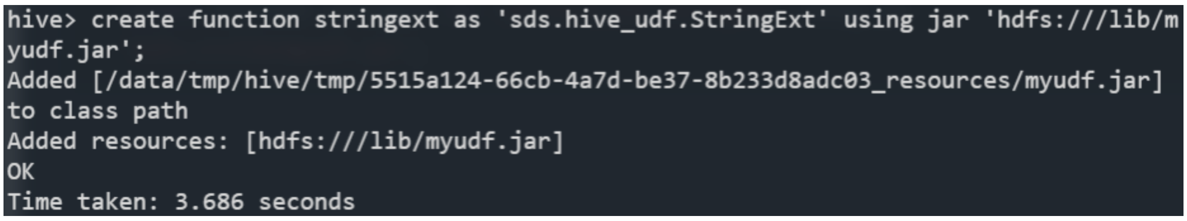
临时函数在hive退出后就会自动失效，永久函数的创建方法时将jar包上传Hdfs。



```
chen@ubuntu:~$ hadoop fs -ls /
Found 6 items
drwxr-xr-x   - chen supergroup          0 2020-11-01 20:01 /hbase
drwxr-xr-x   - chen supergroup          0 2020-11-03 23:58 /home
drwxr-xr-x   - chen supergroup          0 2020-11-03 20:04 /input
drwxr-xr-x   - chen supergroup          0 2020-11-04 00:02 /output
drwx------   - chen supergroup          0 2020-11-03 19:35 /tmp
drwxr-xr-x   - chen supergroup          0 2020-11-03 19:51 /user
chen@ubuntu:~$ hadoop fs -mkdir /lib
chen@ubuntu:~$ hadoop fs -ls /
Found 7 items
drwxr-xr-x   - chen supergroup          0 2020-11-01 20:01 /hbase
drwxr-xr-x   - chen supergroup          0 2020-11-03 23:58 /home
drwxr-xr-x   - chen supergroup          0 2020-11-03 20:04 /input
drwxr-xr-x   - chen supergroup          0 2020-11-09 03:47 /lib
drwxr-xr-x   - chen supergroup          0 2020-11-04 00:02 /output
drwx------   - chen supergroup          0 2020-11-03 19:35 /tmp
drwxr-xr-x   - chen supergroup          0 2020-11-03 19:51 /user
hchen@ubuntu:~$ hadoop fs -put project_jar/myudf.jar /lib/
chen@ubuntu:~$ hadoop fs -ls -R /lib
-rw-r--r--   1 chen supergroup       3525 2020-11-09 03:47 /lib/myudf.jar
```

创建永久函数

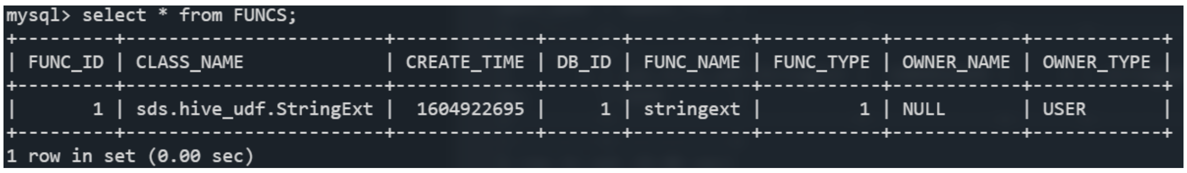
```
create function stringext as 'sds.hive_udf.StringExt' using jar
'hdfs:///lib/myudf.jar';
```



```
hive> create function stringext as 'sds.hive_udf.StringExt' using jar 'hdfs:///lib/m
yudf.jar';
Added [/data/tmp/hive/tmp/5515a124-66cb-4a7d-be37-8b233d8adc03_resources/myudf.jar]
to class path
Added resources: [hdfs:///lib/myudf.jar]
OK
Time taken: 3.686 seconds
```

打开Mysql

查看FUNCS



```
mysql> select * from FUNCS;
+---------+------------------------+-------------+-------+-----------+-----------+------------+------------+
| FUNC_ID | CLASS_NAME             | CREATE_TIME | DB_ID | FUNC_NAME | FUNC_TYPE | OWNER_NAME | OWNER_TYPE |
+---------+------------------------+-------------+-------+-----------+-----------+------------+------------+
|       1 | sds.hive_udf.StringExt |  1604922695 |     1 | stringext |         1 | NULL       | USER       |
+---------+------------------------+-------------+-------+-----------+-----------+------------+------------+
1 row in set (0.00 sec)
```

可以看到函数 stringext 的信息已经注册。