Hive 高级用法

刘磊

2020 年 11 月

目录

1. 表连接

Hive 支持常用的 SQL join 语句，例如内连接、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。

建立两个表 stu 和 score，分别从文件载入数据。

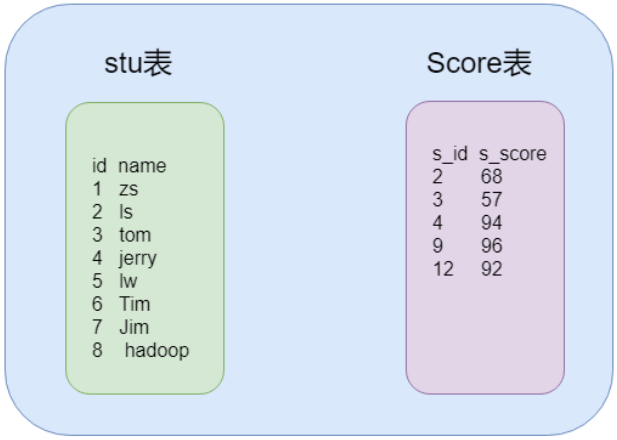
```
create table stu(id int, name string) row format delimited fields
terminated by ',' stored as textfile;
```

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

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

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

两个表的内容如下：



内连接

只有进行连接的两个表中都存在与连接条件相匹配的数据才会被显示。

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

```
2       ls      2       68
3       tom     3       57
4       jerry   4       94
Time taken: 49.316 seconds, Fetched: 3 row(s)
```

左外连接

左外连接是显示左边的表的所有数据，如果有右边表与之对应，则显示；否则显示 NULL。

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

```
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: 63.903 seconds, Fetched: 8 row(s)
```

右外连接

右外连接是显示右边的表的所有数据，如果有左边表与之对应，则显示；否则显示 NULL。

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

```
2       ls      2       68
3       tom     3       57
4       jerry   4       94
NULL    NULL    9       96
NULL    NULL    12      92
Time taken: 50.256 seconds, Fetched: 5 row(s)
```

全外连接

两个表的数据都显示，如果左表或右表中没有对应的数据，则显示 NULL.

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

```
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
```

左半连接

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

```
2       ls
3       tom
4       jerry
Time taken: 44.501 seconds, Fetched: 3 row(s)
```

相当于下面使用 in 的语句

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

注：in：左边的表在右边表的范围内

2. Hive 的连接

2.1 CLI 连接

在命令行直接直接输入命令

```
hive
```

说明：

1. 上面的 hive 命令相当于在启动的时候执行：hive --service cli；
2. 使用 hive --help，可以查看 hive 命令可以启动那些服务；
3. 通过 hive --service serviceName --help 可以查看某个具体命令的使用方式。

```
lei@ubuntu:~$ hive --help
Usage ./hive <parameters> --service serviceName <service parameters>
Service List: beeline cleardanglingscratchdir cli hbaseimport hbaseschematool he
lp hiveburninclient hiveserver2 hplsql jar lineage llapdump llap llapstatus meta
store 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
```

2.2 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。

2.3 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 连接远程服务。

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。

```
<property>
    <name>hive.server2.thrift.client.user</name>
    <value>lei</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>
```

```
4487    <property>
4488      <name>hive.server2.thrift.client.user</name>
4489      <value>lei</value>
4490      <description>Username to use against thrift client</description>
4491    </property>
4492    <property>
4493      <name>hive.server2.thrift.client.password</name>
4494      <value>123456</value>
4495      <description>Password to use against thrift client</description>
4496    </property>
```

在 Hadoop 的配置文件 core-site.xml 增加如下配置，其中 “xxx” 是连接 beeline 的用户，将 “xxx” 替换成自己的用户名和用户所属组即可。hadoop.proxyuser.hadoop.hosts 配置成*的意义，表示任意节点使用 Hadoop 集群的代理用户 xxx 都能访问 HDFS 集群，hadoop.proxyuser.hadoop.groups 表示代理用户 xxx 的所属组在任意节点都能访问 HDFS 集群。

```
<property>
    <name>hadoop.proxyuser.xxx.hosts</name>
    <value>*</value>
```

```
</property>
<property>
    <name>hadoop.proxyuser.xxx.groups</name>
    <value>*</value>
</property>
```

```
    <property>
        <name>hadoop.proxyuser.lei.hosts</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.lei.groups</name>
        <value>*</value>
    </property>
```

不做这个设置会有权限限制的问题，启动 beeline 连接报错： User: xxx is not allowed to impersonate xxx (state=08S01,code=0)。

使用 Beeline 前需要开启 HiveServer2，打开一个终端，运行

```
hiveserver2 &
```

启动会多一个进程 `RunJar`

```
lei@ubuntu:~$ jps
4464 RunJar
2386 DataNode
2611 SecondaryNameNode
2227 NameNode
3076 NodeManager
2922 ResourceManager
4700 Jps
```

登录 Beeline，在另一个终端中运行

```
beeline
```

```
lei@ubuntu:/apps/hive/conf$ 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
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 2.3.5 by Apache Hive
```

连接 Hive2 服务器

```
!connect jdbc:hive2://ubuntu:10000/default
```

按要求输入用户名和密码，出现如下界面说明连接成功

```
beeline> !connect jdbc:hive2://ubuntu:10000/default
Connecting to jdbc:hive2://ubuntu:10000/default
Enter username for jdbc:hive2://ubuntu:10000/default: lei
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>
```

还可以在终端中，执行如下命令登录

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

说明：

- -u: 指定元数据库的链接信息
- -n: 指定用户名
- -p: 指定密码

```
lei@ubuntu:/apps/hive/conf$ beeline -u jdbc:hive2://ubuntu:10000/default -n lei
-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
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
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>
```

登录成功以后就可以对 Hive 进行操作

```
0: jdbc:hive2://ubuntu:10000/default> show databases;
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
1 row selected (1.872 seconds)
```

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，命令前面需要加!，不需要终止符。

!connect url:连接不同的 hiveserver2 服务器。

!exit: 退出 Shell。

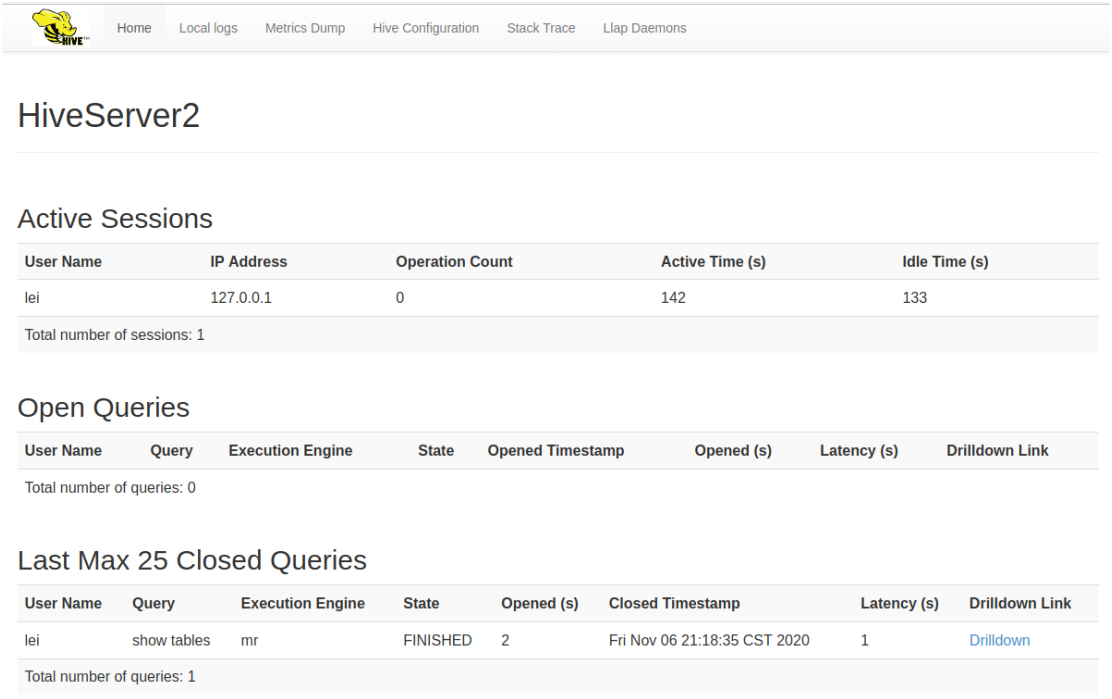
```
0: jdbc:hive2://ubuntu:10000/default> !exit
Closing: 0: jdbc:hive2://ubuntu:10000/default
```

!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 connection
s
!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.
```

2.4 HiveServer2 WEB UI 界面

启动 hiveserver2，在浏览器中打开 http://localhost:10002。在打开的页面中可以看到当前链接的会话、历史日志、配置参数以及度量信息。



3. Hive JDBC

Hive JDBC 类似于 Java 访问关系型数据库的方式，主要是 URL 和驱动不一样，而且主要是查询数据，不能更新和删除数据。
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）

首先，编写 HiveService 类，实现打开连接，断开连接，获取语句等常用方法。

```
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://ubuntu:10000";
```

```
    // 登录 linux 的用户名 一般会给权限大一点的用户，否则无法进行事务形操作
    public static String user = "lei";
    // 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();
        }
    }
}
```

然后创建类 HiveTest，测试 Hive JDBC 创建表，插入数据，加载数据，查询数据，列出全部表，显示表结构，删除表等功能。

```
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);
```

```
            // 需要拥有 hdfs 文件读写权限的用户才可以进行此操作
            stmt.execute("drop table if exists users");
            logger.debug("drop table is susscess");
            // 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 susscess");
            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
            stmt.execute("insert into users(user_id, fname,lname)
values(222,'yang','yang2')");
            logger.debug("insert is susscess");

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

            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);
    }
}
```

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。启动 hiveserver2，运行 HiveTest，结果如下

```
2020-11-06T22:34:50,066 INFO [main] org.apache.hive.jdbc.Utils - Supplied authorities: ubuntu:10000
2020-11-06T22:34:50,070 INFO [main] org.apache.hive.jdbc.Utils - Resolved authority: ubuntu:10000
users.user_id   users.fname     users.lname
1               wu              zy
2               zhang           shan
3               li              si

Running: show tables
iris_bucket
iris_external
iris_flower
iris_partition
iris_partition_bucket
iris_result
```

4. Hive UDF

UDF（User-Defined Functions）是用户定义的 Hive 函数。Hive 自带的函数并不能完全满足业务需求时就需要自定义函数。UDF 函数可以直接应用于 select 语句，对查询结构做格式化处理后，再输出内容。编写 UDF 函数的时候需要注意一下几点：

1) 自定义 UDF 需要继承 org.apache.hadoop.hive.ql.exec.UDF。

2) 需要实现 evaluate 函。

3) 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。

```
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) {

    }
}
```

将工程打包为 myudf.jar（过程见 mr_examples 工程的打包方法）。

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/lei/project_jar/myudf.jar;
```

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

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: 0.673 seconds
```

运行结果

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

```
hive> select fname,stringext(lname) from users;
OK
wu      Hello zy
zhang   Hello shan
li      Hello si
Time taken: 5.382 seconds, Fetched: 3 row(s)
```

可以看出查询的名字前面加了 Hello。

上述方法在 Hive 退出后，自定义的函数就会失效，要是自定义函数不失效，可以将导出的

Jar 包上次到 HDFS，创建永久函数。

上传 jar 包到 HDFS/lib 目录下

```
hadoop fs -put project_jar/myudf.jar /lib
```

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:///l
ib/myudf.jar';
Added [/data/tmp/hive/tmp/26e97476-d613-4b01-b2ef-d987d9da03eb_resources/myudf.j
ar] to class path
Added resources: [hdfs:///lib/myudf.jar]
OK
Time taken: 0.156 seconds
```

打开 MySQL

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

切换到数据库 hive

```
use hive;
```

查看表 FUNCS

```
select * from FUNCS;
```

```
mysql> select * from FUNCS;
+---------+-----------------------+-------------+-------+-----------+----------
-+------------+------------+
| FUNC_ID | CLASS_NAME            | CREATE_TIME | DB_ID | FUNC_NAME | FUNC_TYPE
 | OWNER_NAME | OWNER_TYPE |
+---------+-----------------------+-------------+-------+-----------+----------
-+------------+------------+
|       2 | sds.hive_udf.StringExt |  1604722833 |     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。