Day10_Hive的安装及入门

大数据-张军锋 Day10 安装 文档 应用

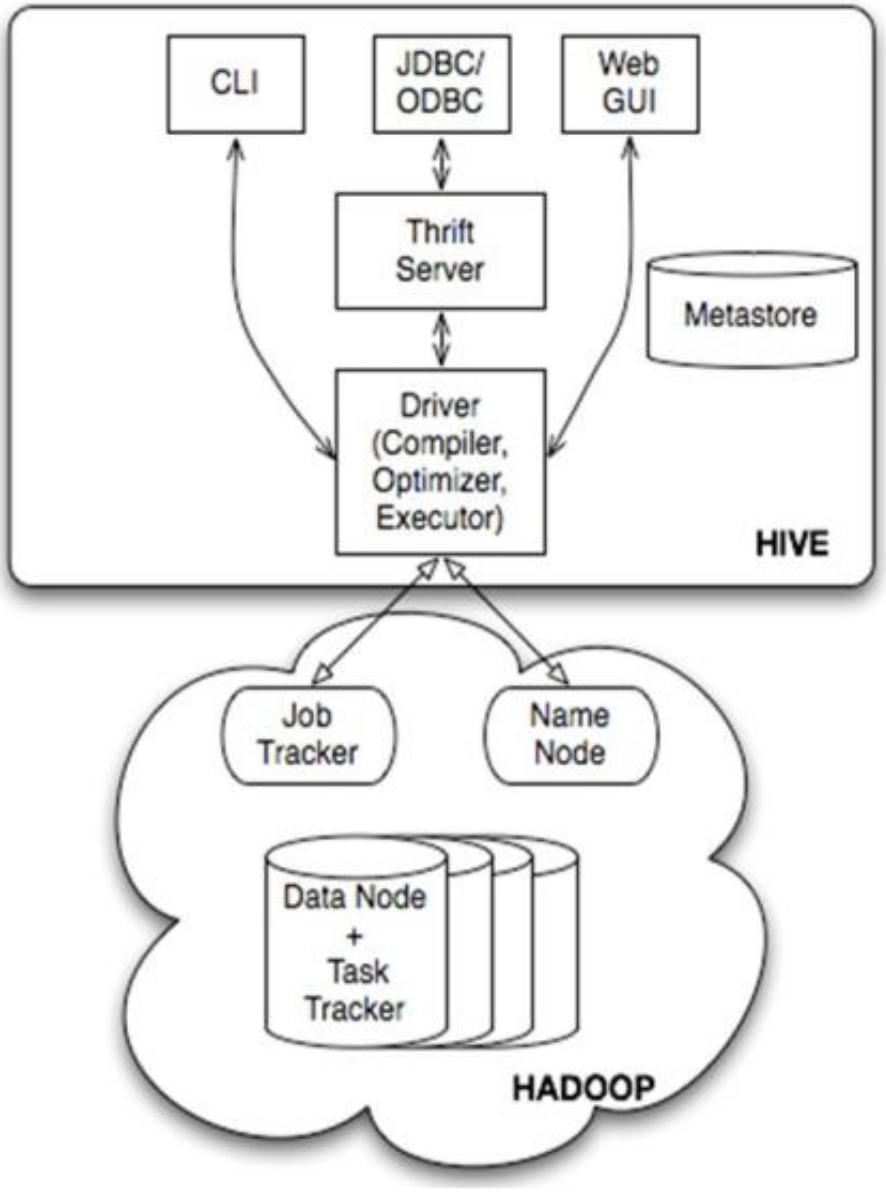
Day10_Hive的安装及入门
- Hive简介
  - 结构
  - Hive架构
  - Hive的组成
    - Hive 的核心
    - Hive 的底层存储
    - Hive 程序的执行过程
    - Hive 的元数据存储
    - Hive的客户端
  - Hive与Hadoop的关系
  - Hive 和普通关系数据库的异同
- Hive的安装
  - 安装Hive
  - 安装Mysql作为Hive的Metastore
  - 配置Hive
- 常用Hive操作数据库语句
- SQuirreL SQL Client使用入门
  - SQuirreL SQL Client简介
  - SQuirreL SQL Client的下载及安装
  - 写一个简单的Helloword玩一玩
- Hive中的数据类型
  - 数字类型
  - 日期/时间类型
  - 字符串类型
  - 杂项类型
  - 复杂类型
- DQL、DML、DDL、DCL的概念与区别

Hive简介

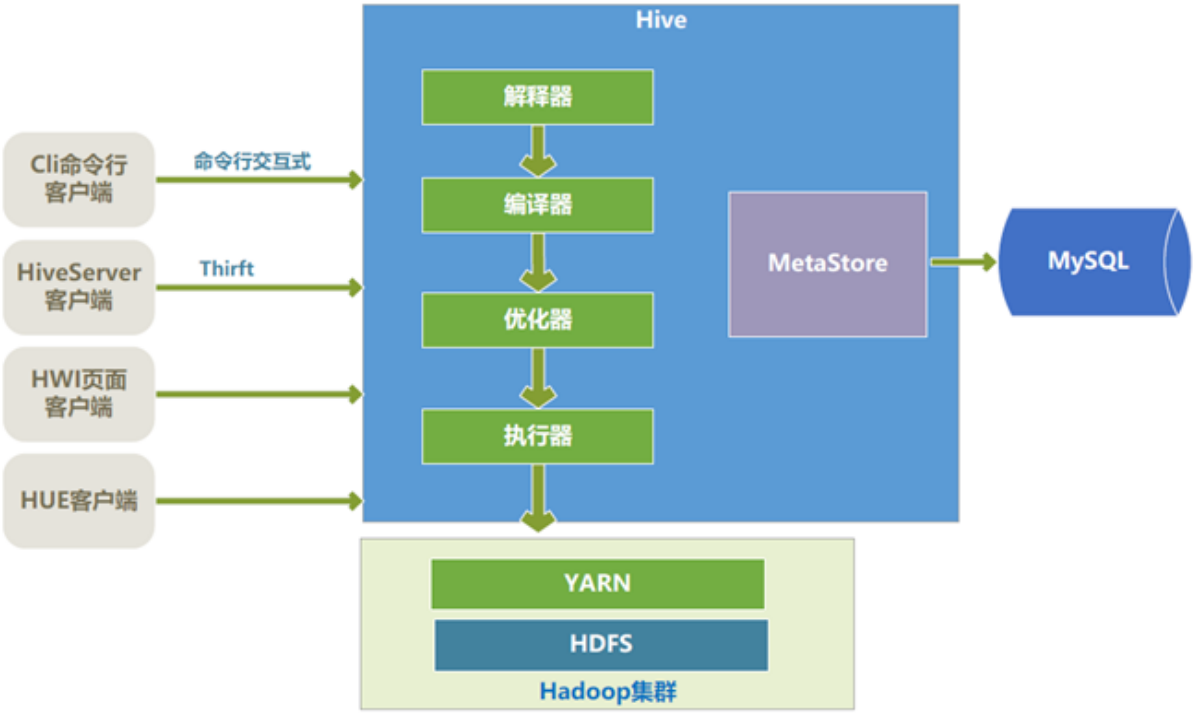
结构

Hive 是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 hQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的

Hive架构



Hive的组成



Hive 的核心

Hive 的核心是Driver驱动引擎，驱动引擎由四部分组成：

1. 解释器：解释器的作用是将HiveSQL 语句转换为语法树（AST）。
2. 编译器：编译器是将语法树编译为逻辑执行计划。
3. 优化器：优化器是对逻辑执行计划进行优化。
4. 执行器：执行器是调用底层的运行框架执行逻辑执行计划。

Hive 的底层存储

Hive 的数据是存储在HDFS 上的。Hive 中的库和表可以看做是对HDFS 上数据做的一个映射。所以Hive 必须是运行在一个Hadoop 集群上的。

Hive 程序的执行过程

Hive 中的执行器，是将最终要执行的MapReduce 程序放到YARN 上以一系列Job 的方式去执行。

Hive 的元数据存储

Hive 的元数据是一般是存储在MySQL 这种关系型数据库上的，Hive 和MySQL 之间通过MetaStore服务交互。

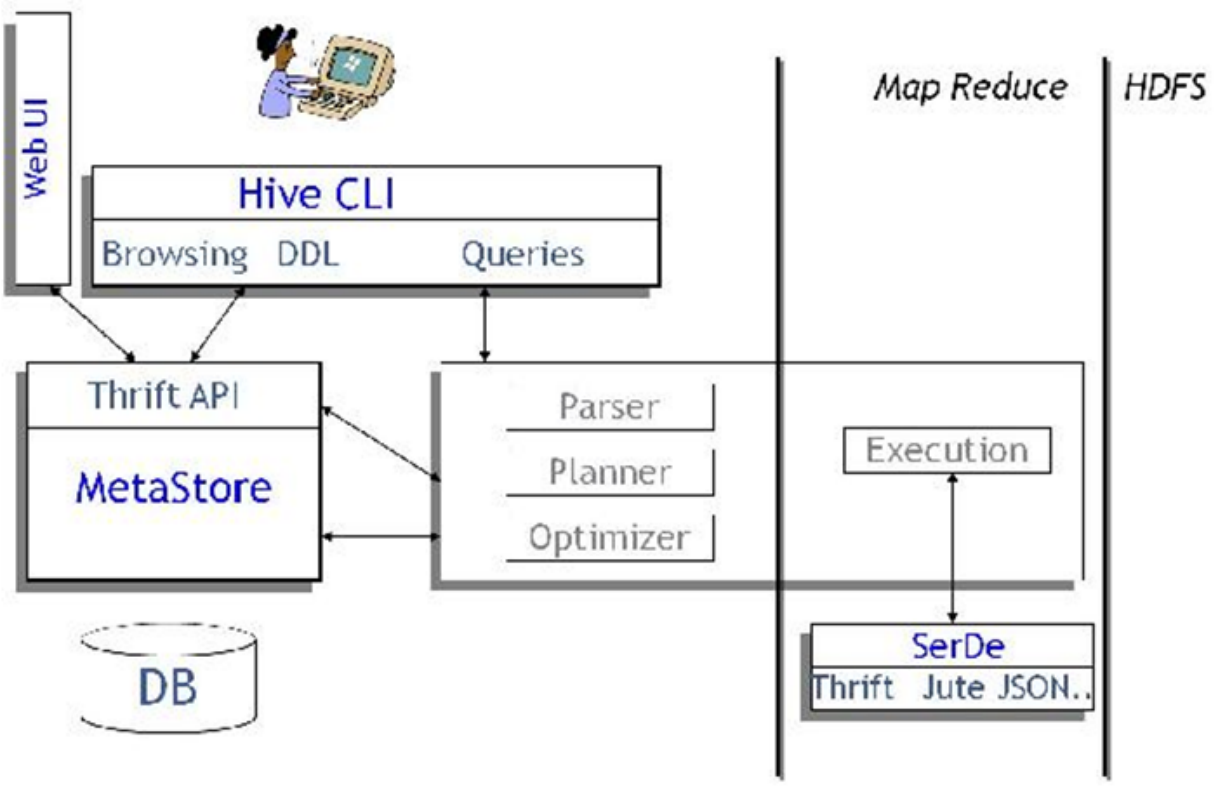
> Metastore是元数据库，里面保存的是Hive的数据模式，Hive中表和分区的所有元数据都存储在Metastore中。

Owner	库、表的所属者	CreateTime	创建时间
LastAccessTime	最后修改时间	Location	存储位置
Table Type	表类型（内部表、外部表）		表的字段信息

Hive的客户端

1. **cli 命令行客户端：**采用交互窗口，用hive 命令行和Hive 进行通信。
2. **HiveServer2 客户端：**用Thrift 协议进行通信，Thrift 是不同语言之间的转换器，是连接不同语言程序间的协议，通过JDBC 或者ODBC 去访问Hive。
3. **HWI 客户端：**hive 自带的一个客户端，但是比较粗糙，一般不用。
4. **HUE 客户端：**通过Web 页面来和Hive 进行交互，使用的比较多。

Hive与Hadoop的关系



- HQL 中对查询语句的解释、优化、生成查询计划是由 Hive 完成的
- 所有的数据都是存储在 Hadoop 中
- 查询计划被转化为 MapReduce 任务，在 Hadoop 中执行（有些查询没有 MR 任务，如：select * from table）
- Hadoop和Hive都是用UTF-8编码的

Hive 和普通关系数据库的异同

Item	Hive	RDBMS
查询语言	HQL	SQL
数据存储	HDFS	Raw Device or Local FS
索引	有	有
执行	MapReduce	Excutor
执行延迟	高	低
处理数据规模	大	小

查询语言：由于 SQL 被广泛的应用在数据仓库中，因此，专门针对 Hive 的特性设计了`类 SQL 的查询语言 HQL`。熟悉 SQL 开发的开发者可以很方便的使用 Hive 进行开发。

数据存储位置：Hive 是建立在 Hadoop 之上的，所有 Hive 的数据都是存储在`HDFS`中的。而数据库则可以将数据保存在块设备或者本地文件系统中。

数据格式：Hive 中没有定义专门的数据格式，数据格式可以由用户指定，用户定义数据格式需要指定三个属性：`列分隔符`（通常为空格、”\t”、”\x001″）、`行分隔符`（”\n”）以及`读取文件数据的方法`（Hive 中默认有三个文件格式 TextFile，SequenceFile 以及 RCFile）。

> 由于在加载数据的过程中，不需要从用户数据格式到 Hive 定义的数据格式的转换，因此，Hive 在加载的过程中不会对数据本身进行任何修改，而只是将数据内容复制或者移动到相应的 HDFS 目录中。而在数据库中，不同的数据库有不同的存储引擎，定义了自己的数据格式。所有数据都会按照一定的组织存储，因此，数据库加载数据的过程会比较耗时。

数据更新：由于 Hive 是针对数据仓库应用设计的，而数据仓库的内容是`读多写少`的。因此，`Hive 中不支持对数据的改写和添加`，所有的数据都是在加载的时候中确定好的。而数据库中的数据通常是需要经常进行修改的，因此可以使用`INSERT INTO ... VALUES`添加数据，使用`UPDATE ... SET`修改数据。

索引：之前已经说过，Hive 在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些 Key 建立索引。Hive 要访问数据中满足条件的特定值时，需要暴力扫描整个数据，因此访问延迟较高。由于 MapReduce 的引入， Hive 可以并行访问数据，因此即使没有索引，对于大数据量的访问，Hive 仍然可以体现出优势。数据库中，通常会针对一个或者几个列建立索引，因此对于少量的特定条件的数据的访问，数据库可以有很高的效率，较低的延迟。由于数据的访问延迟较高，决定了`Hive 不适合在线数据查询`。

执行：Hive 中大多数查询的执行是通过 Hadoop 提供的 MapReduce 来实现的（类似 select * from tbl 的查询不需要 MapReduce）。而数据库通常有自己的执行引擎。

执行延迟：之前提到，Hive 在查询数据的时候，由于没有索引，需要扫描整个表，因此延迟较高。另外一个导致 Hive 执行延迟高的因素是 MapReduce 框架。由于 MapReduce 本身具有较高的延迟，因此在利用 MapReduce 执行 Hive 查询时，也会有较高的延迟。相对的，数据库的执行延迟较低。当然，这个低是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时候，Hive 的并行计算显然能体现出优势。

可扩展性：由于 Hive 是建立在 Hadoop 之上的，因此 Hive 的可扩展性是和 Hadoop 的可扩展性是一致的（世界上最大的 Hadoop 集群在 Yahoo!，2009年的规模在 4000 台节点左右）。而数据库由于 ACID 语义的严格限制，扩展行非常有限。目前最先进的并行数据库 Oracle 在理论上的扩展能力也只有 100 台左右。

数据规模：由于 Hive 建立在集群上并可以利用 MapReduce 进行并行计算，因此可以支持很大规模的数据；对应的，数据库可以支持的数据规模较小。

Hive的安装

安装Hive

安装在hadoop的namenode上，拷贝安装文件到linux中/opt/Software/Hive/apache-hive-2.0.0-bin.tar.gz
解压： `tar -zxvf apache-hive-2.3.0-bin.tar.gz`
添加到环境变量 `vi /etc/profile`

```
#hive
export HIVE_HOME=/opt/Software/Hive/apache-hive-2.3.0-bin
export PATH=$PATH:$HIVE_HOME/bin
```

保存后使其生效： `source /etc/profile`

安装Mysql作为Hive的Metastore

首先检查mysql是否已安装： `rpm -qa | grep -i mysql`
结果： `mysql-libs-5.1.71-1.el6.x86_64`
删除已安装 `mysql yum -y remove mysql-libs*`

解压： `tar xvf MySQL-5.5.49-1.linux2.6.x86_64.rpm-bundle.tar`
安装： `rpm -ivh MySQL-server-5.5.49-1.linux2.6.x86_64.rpm`
`rpm -ivh MySQL-devel-5.5.49-1.linux2.6.x86_64.rpm`
`rpm -ivh MySQL-client-5.5.49-1.linux2.6.x86_64.rpm`

启动mysql： `service mysql start`
设置root用户登录密码：首次安装时，默认密码为空，可以使用如下命令修改root密码
`mysqladmin -u root password mypassword`
mypassword 为你设定的新密码
登录mysql： `mysql -uroot -proot`

rpm包安装的MySQL是不会安装/etc/my.cnf文件的
解决方法：只需要复制/usr/share/mysql目录下的my-huge.cnf 文件到/etc目录，并改名为my.cnf即可 `cp /usr/share/mysql/my-huge.cnf /etc/my.cnf`

mysql默认不可以远程访问，**设置远程访问**
免密码访问： `GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'WITH GRANT OPTION;`
需要密码访问： `GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'IDENTIFIED BY 'root' WITH GRANT OPTION;`
使权限生效： `FLUSH PRIVILEGES;`

设置etc/my.cnf文件：使binlog_format=mixed
`vi etc/my.cnf'`
将注释掉的binlog_format=mixed这一行前面的注释去掉然后保存，重启mysql即可
`service mysql restart`

配置Hive

在hdfs中新建目录/user/hive/warehouse

```
hdfs dfs -mkdir /tmp
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hive
hdfs dfs -mkdir /user/hive/warehouse

hadoop fs -chmod g+w /tmp
hadoop fs -chmod g+w /user/hive/warehouse
```

将mysql的驱动jar包mysql-connector-java-5.1.7-bin.jar拷入hive的lib目录下面

修改hive-site.xml
进入hive的conf目录下面复制一下hive-default.xml.template名字命名为：hive-site.xml
`cp hive-default.xml.template hive-site.xml`
然后修改hive-site.xml

```
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://127.0.0.1:3306/hive?createDatabaseIfNotExist=tr
ue</value>
<description>JDBC connect string for a JDBC metastore</description>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
<description>Username to use against metastore database</descriptio
n>
</property>
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>root</value>
<description>password to use against metastore database</descriptio
n>
</property>

<property>
<name>hive.exec.local.scratchdir</name>
<value>/opt/Software/Hive/apache-hive-2.3.0-bin/tmp</value>
<description>Local scratch space for Hive jobs</description>
</property>
<property>
<name>hive.downloaded.resources.dir</name>
<value>/opt/Software/Hive/apache-hive-2.3.0-bin/tmp/resources</valu
e>
<description>Temporary local directory for added resources in the r
emote file system.</description>
</property>
<property>
<name>hive.querylog.location</name>
<value>/opt/Software/Hive/apache-hive-2.3.0-bin/tmp</value>
<description>Location of Hive run time structured log file</descrip
tion>
</property>
<property>
<name>hive.server2.logging.operation.log.location</name>
<value>/opt/Software/Hive/apache-hive-2.3.0-bin/tmp/operation_log
s</value>
<description>Top level directory where operation logs are stored if
logging functionality is enabled</description>
```

```
  </property>
```

使用schematool 初始化metastore的schema：
`schematool -initSchema -dbType mysql`

运行Hive： `hive`

常用Hive操作数据库语句

> 一个简单的WordCount，包含一些数据库的操作，就不分开整理了。

- **启动Hive服务：** `hiveserver2`
- 再打开一个窗口，**使用Beeline客户端：** `beeline`
- **建立连接：** `!connect jdbc:hive2://master:10000/` 按照提示输入账户，密码（设置免密登录，直接Enter）
- **创建数据库：** `create database bd14;`
- **使用数据库：** `use bd14;`
- **查看当前数据库中所有的表：** `show tables;`
- **删除数据库：** `drop database bd14;`
- **删除存在表的数据库：** `drop database bd14 cascade;`
- **创建一个表，字段之间用 \t 分隔：** `create table student (id int, name string) row format delimited fields terminated by '\t' ;`
- **加载HDFS中的文件到表中：** `load data inpath '/README.txt' overwrite into table docs;`
- **加载本地文件到表中：** `load data local inpath '/home/student.txt' into table student;`
- **查询表内容：** `select * from student;`
- **只查询前两条：** `select * from student limit 2;`
- **查询频度排名（频度最高的前50）：** `select keyword,count(*) as cnt from sogou_1w group by keyword order by cnt desc limit 50;`
- **修改列名：** `alter table test column ·stuname· name string;“` ·是右上角的~键 `describe test;`
- **增加列：** `alter table test add columns(height int); describe test;`
- **替换列：** `alter table test replace columns( id int, name string, age int);`
- **创建外部表：** `create external table ext_student (id int, name string) row format delimited fields terminated by '\t' location '/data';` 这样就不必将文件放到hive里去 就可以对其进行操作了，只需要将文件放到hdfs上的/data目录下面。

> 内部表先有表后有数据；外部表先有数据后有表。

- **创建分区表：** `create external table beauties (id bigint, name string, size double) partitioned by (nation string) row format delimited fields terminated by '\t' location '\beauty';`
- **多表关联：** `select t . account , u . name , t . income , t . expenses , t . surplus from user_info u join (select account , sum(income) as income , sum(expenses) as expenses , sum(income-expenses) as surplus from trade_detail group by account) t on u . account = t . account;`
- **分割字符串：** `select explode(split(line,'\\s+')) from docs;`
- **分割字符串进行WordCount：** `select word,count(*) from (select explode(split(line,'\\s+'))as word from docs) a group by word;`
- **创建一个新表接收查询结果：** `create table wc_count as select word,count(*) from (select explode(split(line,'\\s+'))as word from docs) a group by word;`
- **创建带有分区的内部表：** `create table testpar( id int, name string,age int) PARTITIONED BY (day string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' location '/testpar';`
- **为带有分区的内部表加载数据：** `load data local inpath '/home/test' into table testpar partition (day='0925');`
- **添加防止删除的保护：** `alter table testpar partition (day='0925') enable no_drop;`
- **测试：删除分区** `alter table testpar drop if exists partition (day='0925');`
- **删除添加的”删除”保护：** `alter table testpar partition (day='0925') disable no_drop;`
- **添加防止查询的保护：** `alter table testpar partition (day='0925') enable offline;`
- **删除防止查询的保护：** `alter table testpar partition (day='0925') disable offline;`
- **按id降序排序：** `select * from student order by id desc;`

数学函数

- **int类型rank加运算：** `select rank+1 from student limit 100;`
- **对int字段平方：** `select pow(rank,2) from student;`
- **取模：(如:2对三取模)** `select pmod(2,3) from student limit 10;`

聚合函数

- **统计一个表的行数：** `select count(*) from mytable;`
- **求一个表id字段的id 之和：** `select sum(id) from mytable;`
- **去重查询：**group by的使用 `select * from mytable group by uid;`
- **独立UID总数：** `select count(distinct(uid)) from mytable;` （高效）或者

`hive>select count(*) from(select * from mytable group by uid) a;`

- **最大值&最小值：** `select max(rank), min(rank) from mytable;`
- **强转：** `select cast(rank as DOUBLE) from mytable limit 10;`
- **拼接：** `select concat(uid,url) from mytable limit 10;`

JSON

- **抽取JSON对象的某一属性值：** `select get_json_object('{"name":"xiaoming","age":"15"}','$.age') from mytable limit 5;` 结果：15
  `select get_json_object(channel,'$.age') from ext_sogou_20111230 limit 3;`
- **查找url字符串中的5位置之后字符串baidu第一次出现的位置：** `select locate("baidu",url,5) from mytable limit 100;`
- **抽取字符串baidu中符合正则表达式url的第5个部分的子字符串：** `select regexp_extract("baidu",url,5) from mytable limit 100;`
- **按照正则表达式”0”分割字符串uid,并将分割后的部分以字符串数组的方式返回：** `select split(uid,"0") from mytable limit 100;`
  结果之一：[“”,”875edc8a14a228”,”1bac1ddc”,”1fa18a1”]
- **对字符串url,从0处开截取长度为3的字符串,作为其子字符串：** `select substr(url,0,3) from mytable limit 3;`
- **将字符串url中所有的字母转换成大写字母：** `select upper(url) from mytable limit 3;`

> 提示：如果出现内存错误，去检查一下hadoop的配置文件。

SQuirreL SQL Client使用入门

SQuirreL SQL Client简介

如果您的工作要求您在一天之中连接许多不同的数据库（oracle、DB2、mysql、postgresql、Sql Server等等），或者你经常需要在多个不同种类的数据库之间进行数导入导出。那么SQuirreL SQL Client 将会是比较理想的数据库客户端链接工具。

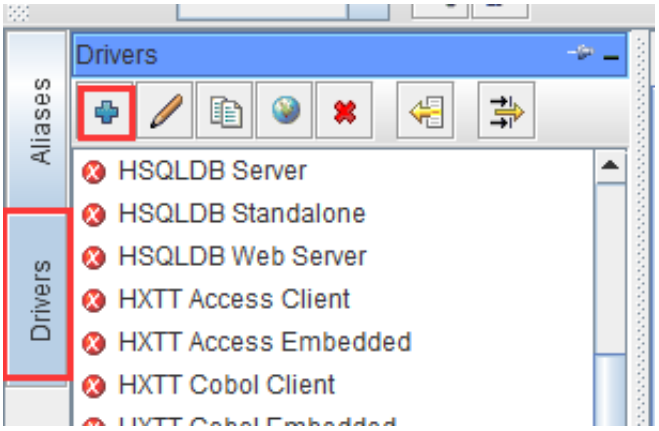
> SQuirrel SQL Client是一个用Java写的数据库客户端，用JDBC统一数据库访问接口以后，可以通过一个统一的用户界面来操作MySQL PostgreSQL MSSQL Oracle等等任何支持JDBC访问的数据库。使用起来非常方便。而且，SQuirrel SQL Client还是一个典型的Swing程序，也算是Swing的一个比较成功的应用了。

SQuirreL SQL Client的下载及安装

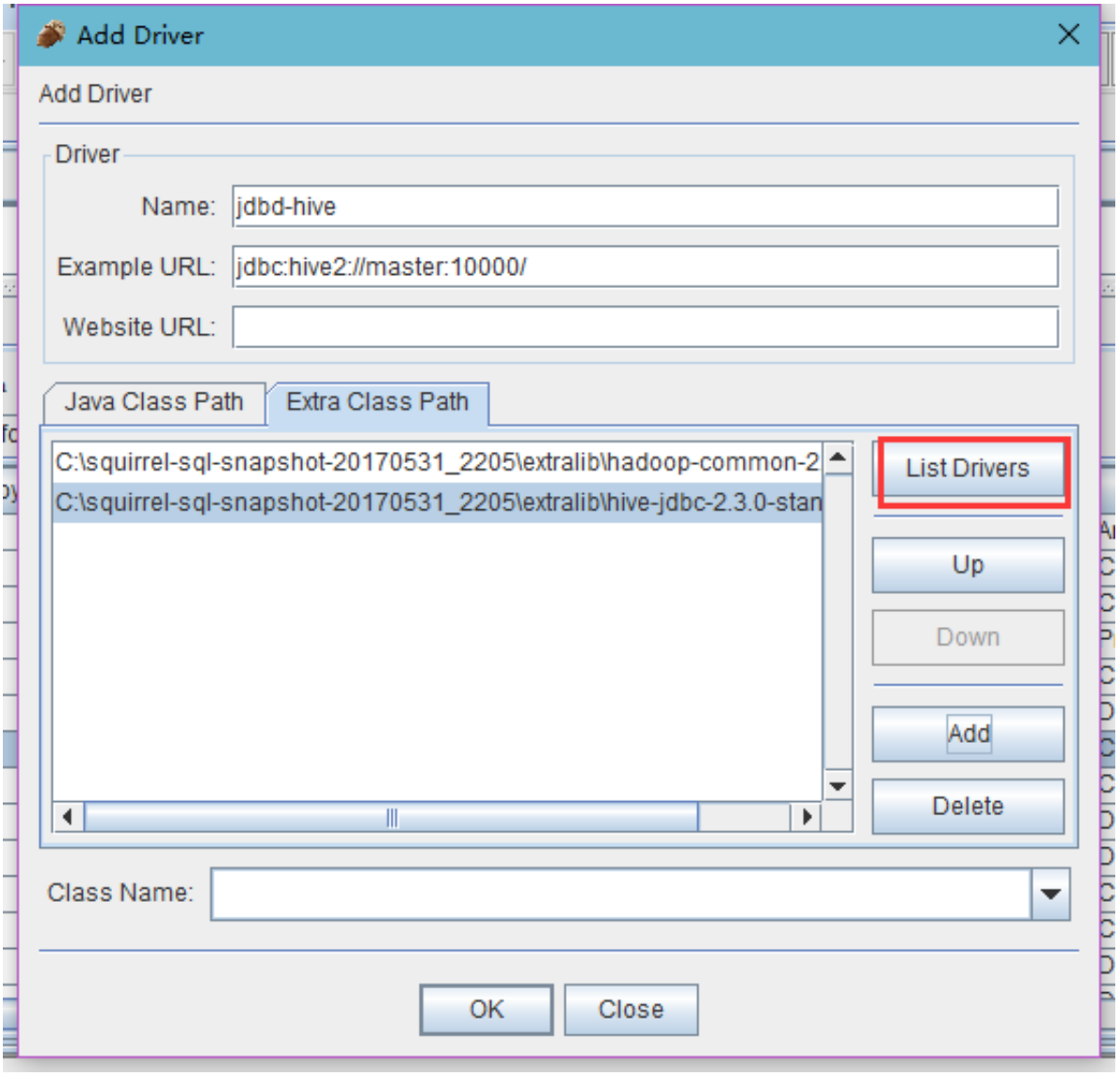
SQuirreL SQL Client下载路径：http://squirrel-sql.sourceforge.net/#installation

安装过程：一路next，安装目录必须是已存在文件夹（期间也可以选择下载需要的驱动包，本次自己手动导入）
导入jar包：在SQuirreL SQL Client的安装目录下新建文件夹extralib
把/opt/Software/Hive/apache-hive-2.3.0-bin/jdbc/**hive-jdbc-2.3.0-standalone.jar**
和/opt/Software/Hadoop/hadoop-2.7.3/share/hadoop/common
/**hadoop-common-2.7.3.jar**拷贝到新建的文件夹中

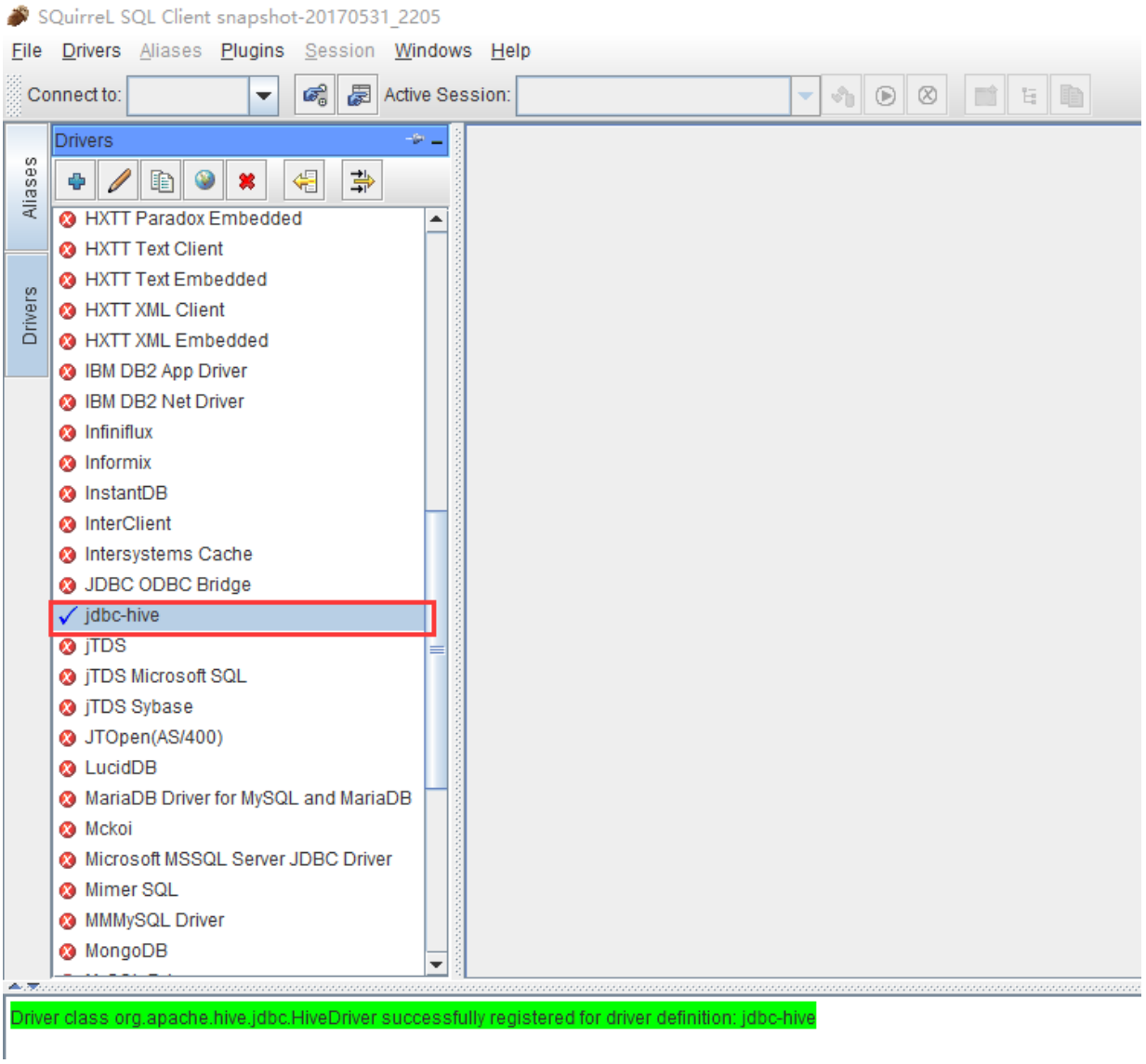
打开squirrel，选择Drivers，点击加号



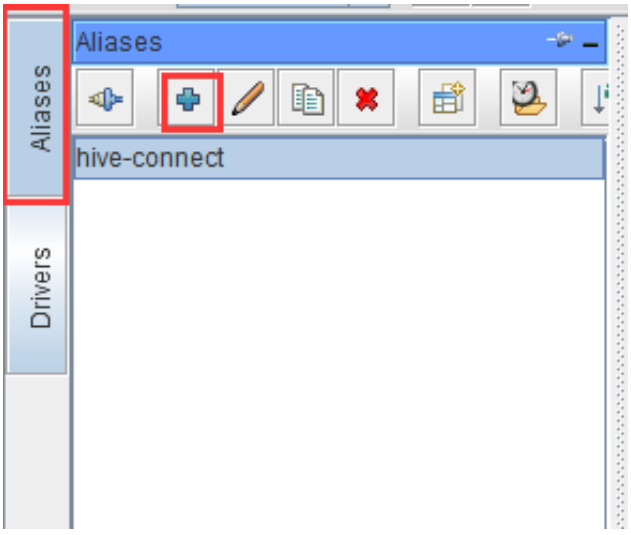
填写Name和Example URL（注意后面要加/），**然后选择Extra Class Path，add刚才拷贝的两个jar包，两个都要List Drivers**



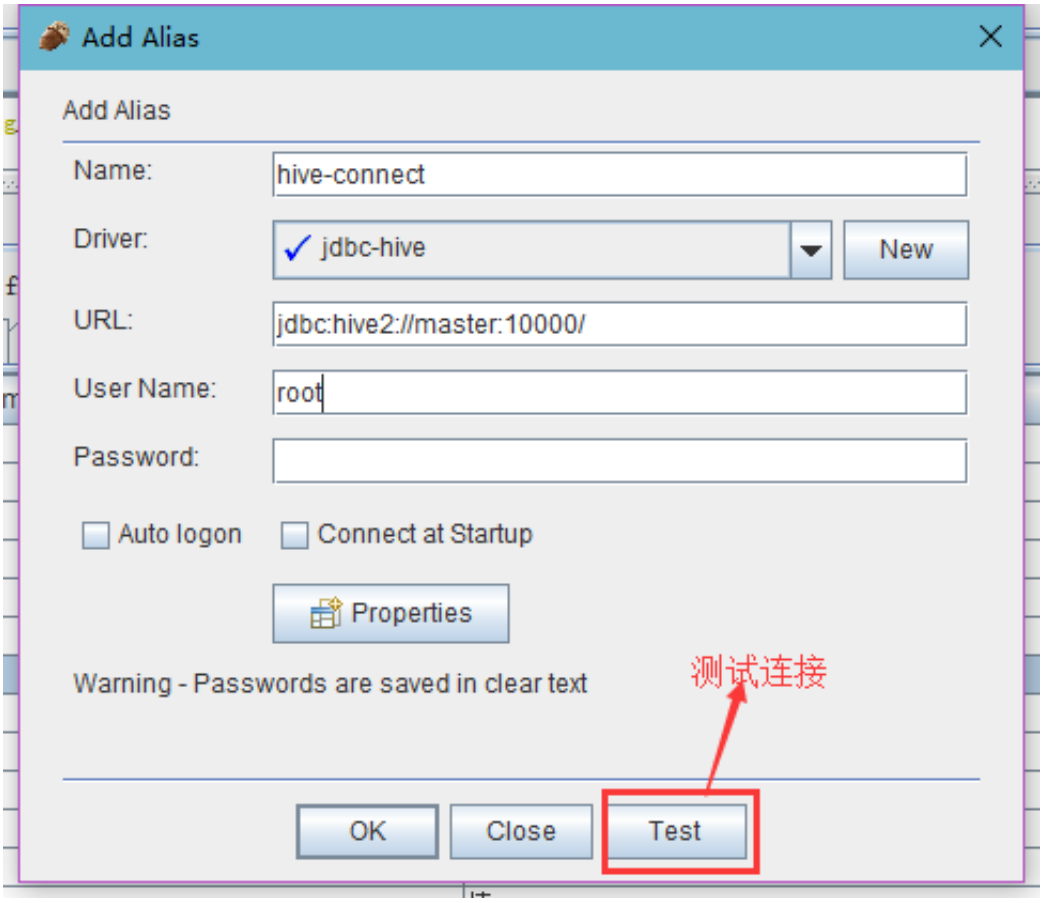
完成之后会发现Drivers里面出现了刚才设置的



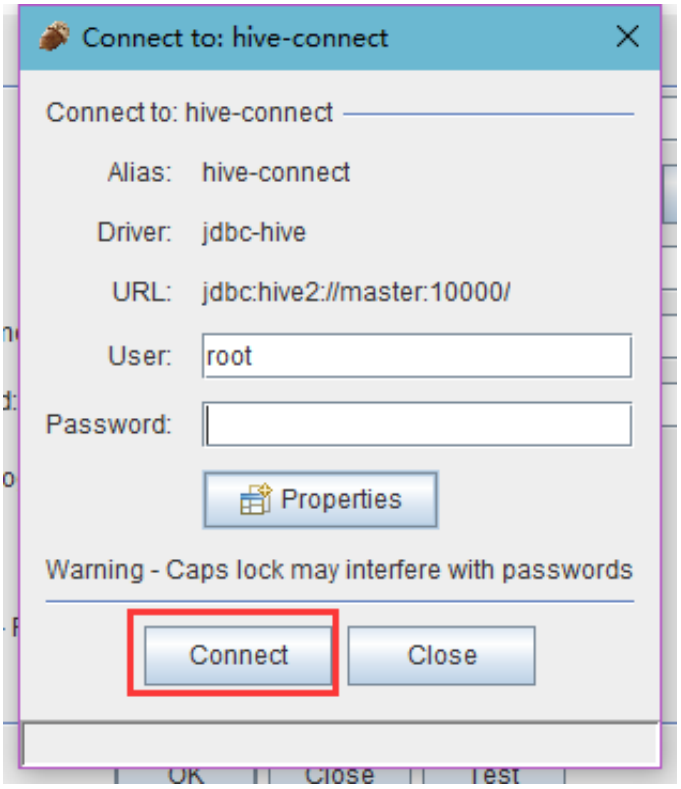
选择Aliases，点击加号

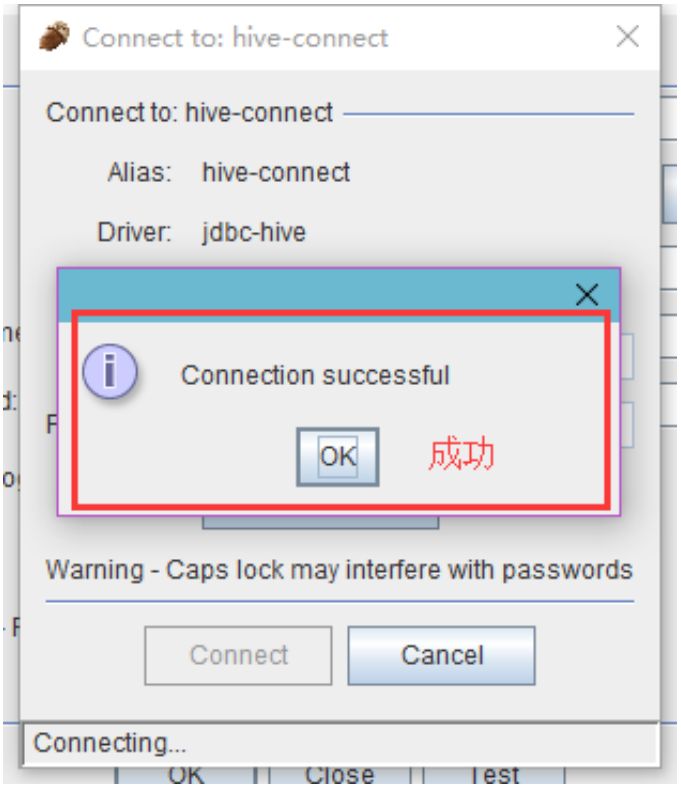


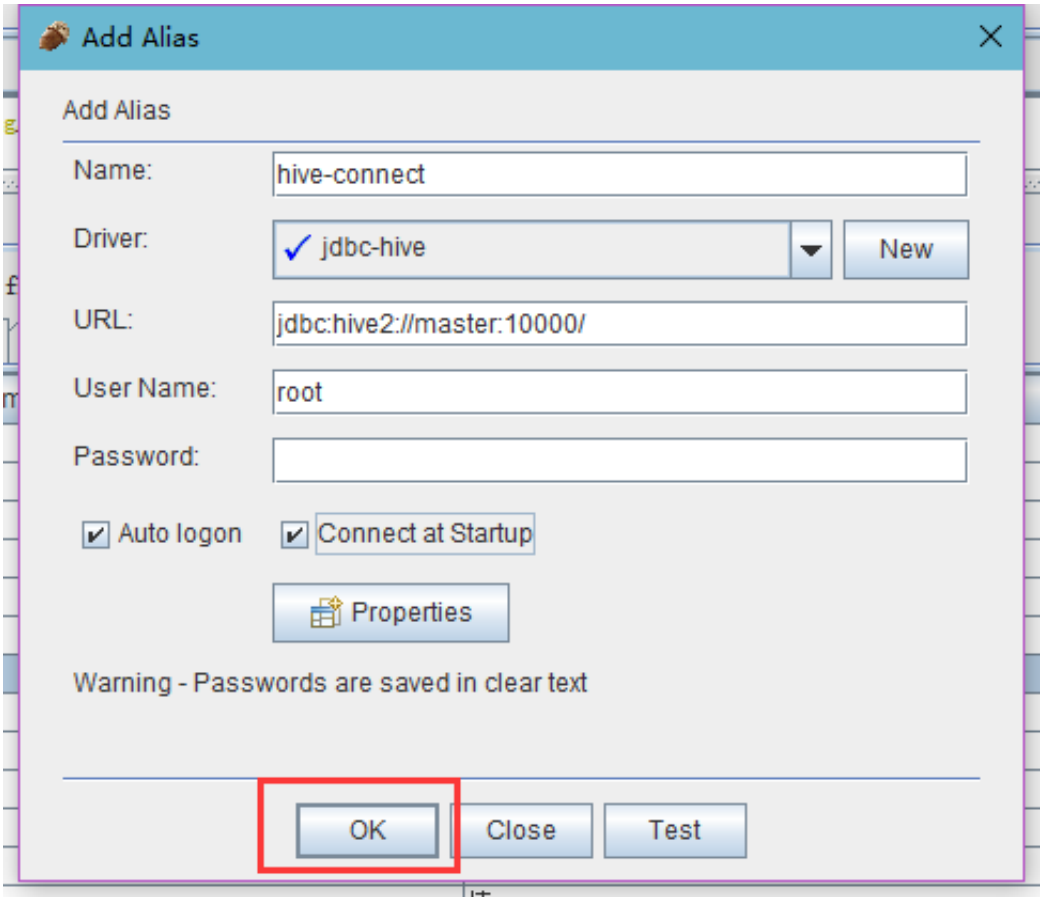
填写用户名（如果设置免密访问，就不需要输入密码），**打开测试连接**



点击Connect，测试成功





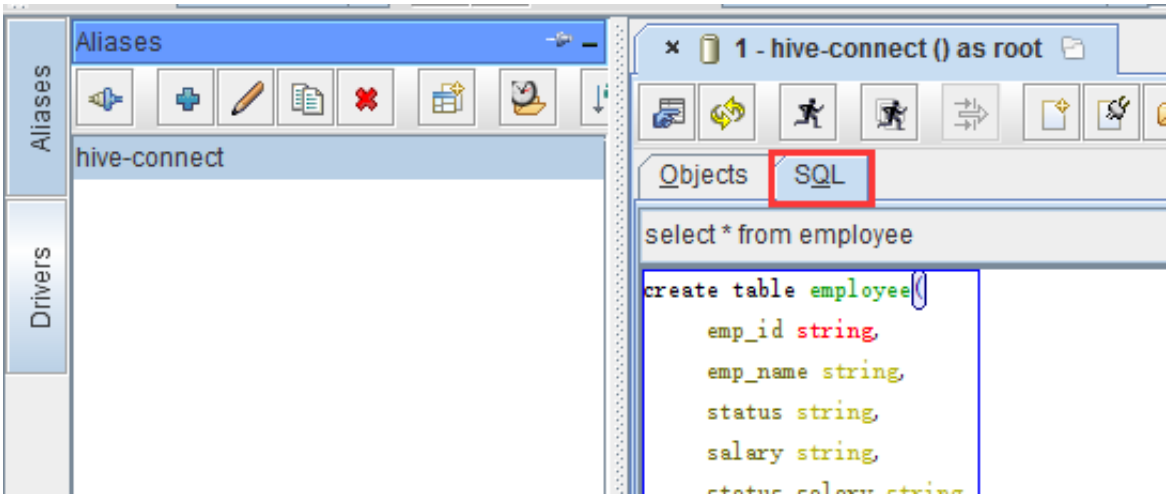


写一个简单的Helloword玩一玩

我们有一个这样的文档，把它上传到HDFS上

```
1000  艾克  Anaylst 7000  2000  2015-01-02  null  1000
1002  亚索  Coder 7030  1000  2014-01-02  1000  1000
1003  蛮王  Coder 6000  null  2015-03-02  1000  1000
1004  剑圣  Programmer  9400  null  2012-01-08  1003  1000
1005  盖伦  Coder 7000  2050  2013-01-06  1003  1000
1006  卡萨丁  Dba 10030 1000  2014-03-02  null  1001
1007  妖姬  Coder 12000 null  2012-03-02  1006  1001
1008  狐狸  Coder 3480  null  2013-07-08  1006  1001
1009  劫  Dba 5000  2050  2013-01-06  1006  1001
1010  凯南  Dba 10030 1000  2014-10-02  null  1002
1011  阿卡丽  Coder 20000 null  2014-03-02  1010  1003
1012  李青  Coder 12080 30  2011-09-08  1010  1004
1013  慎  Dba 5000  2050  2013-11-06  1010  1004
1014  大树  Dba 10030 1000  2014-10-02  null  1002
1015  鳄鱼  Coder 9000  null  2014-03-02  1014  1003
1016  刀妹  Coder 8080  30  2011-09-08  1014  1004
1017  扎克  Dba 7260  2050  2013-11-06  1016  1004
人员编号  人员名称  职位  薪资  岗位薪资  入职时间  上级编号  归属部门编号
```

看到那两个选项卡Objects和SQL,**切换到SQL**



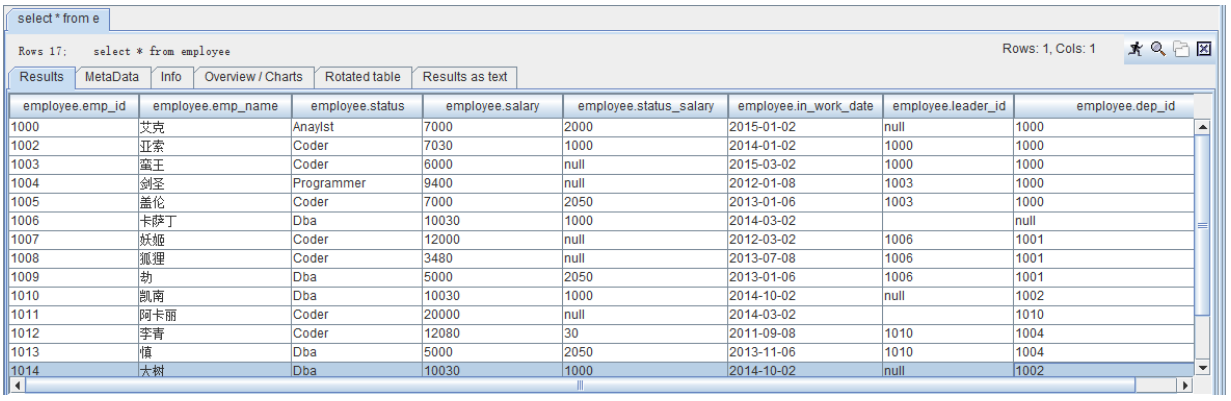
然后切换至准备操作的数据库：`usr bd14;`
紧接着就是建表，导数据一顿操作

```
create table employee(
    emp_id string,
    emp_name string,
    status string,
    salary string,
    status_salary string,
    in_work_date string,
    leader_id string,
    dep_id string
)
row format delimited
fields terminated by '\t'
stored as textfile;

load data inpath '/emp_dep/employee.txt' overwrite into table emplo
yee

select * from employee
```

Show Time



select * from e

Rows 17: select * from employee

Rows: 1, Cols: 1

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

employee.emp_id	employee.emp_name	employee.status	employee.salary	employee.status_salary	employee.in_work_date	employee.leader_id	employee.dep_id
1000	艾克	Anaylst	7000	2000	2015-01-02	null	1000
1002	亚索	Coder	7030	1000	2014-01-02	1000	1000
1003	盖王	Coder	6000	null	2015-03-02	1000	1000
1004	剑圣	Programmer	9400	null	2012-01-08	1003	1000
1005	盖伦	Coder	7000	2050	2013-01-06	1003	1000
1006	卡萨丁	Dba	10030	1000	2014-03-02		null
1007	妖姬	Coder	12000	null	2012-03-02	1006	1001
1008	狐狸	Coder	3480	null	2013-07-08	1006	1001
1009	劫	Dba	5000	2050	2013-01-06	1006	1001
1010	凯南	Dba	10030	1000	2014-10-02	null	1002
1011	阿卡丽	Coder	20000	null	2014-03-02		1010
1012	李青	Coder	12080	30	2011-09-08	1010	1004
1013	慎	Dba	5000	2050	2013-11-06	1010	1004
1014	大树	Dba	10030	1000	2014-10-02	null	1002

Hive中的数据类型

数字类型

TINYINT（1字节有符号整数，从-128到127）
SMALLINT（2字节有符号整数，从-32,768到32,767）
INT / INTEGER（4字节有符号整数，从-2,147,483,648到2,147,483,647）
BIGINT（8字节有符号整数，从-9,223,372,036,854,775,808到9,223,372,036,854,775,807）
FLOAT（4字节单精度浮点数）
DOUBLE（8位双精度浮点数）
DOUBLE PRECISION（ DOUBLE的别名，仅供 Hive 2.2.0开始使用）
DECIMAL（在Hive 0.11.0中引入了38位精度，Hive 0.13.0引入了用户定义的精度和尺度）
NUMERIC（相同DECIMAL，从Hive 3.0.0开始）

日期/时间类型

TIMESTAMP（注：只有从Hive 0.8.0开始）
DATE（注：只有从Hive 0.12.0开始）
INTERVAL（注：只有从Hive 1.2.0开始）

字符串类型

STRING（老朋友，再正常不过的的字符串）
VARCHAR（注：只有从Hive 0.12.0开始）
CHAR（注：只有从Hive 0.13.0开始）

杂项类型

BOOLEAN（还是那个布尔）
BINARY（注：只有从Hive 0.8.0开始）

复杂类型

arrays： `ARRAY<data_type>`（注：负数值和非常数表达式允许为 Hive 0.14。）
maps： `MAP<primitive_type, data_type>`（注：负数值和非常数表达式允许为 Hive 0.14。）
structs： `STRUCT<col_name : data_type [COMMENT col_comment], ...>`
union： `UNIONTYPE<data_type, data_type, ...>`（注：只有从Hive 0.7.0开始。）

DQL、DML、DDL、DCL的概念与区

别

数据定义语言DDL：用来创建数据库中的各种对象——表、视图、索引、同义词、聚簇等。
数据操纵语言DML：主要有三种形式：1) 插入：INSERT；2) 更新：UPDATE；3) 删除：DELETE
数据查询语言DQL：基本结构是由SELECT子句，FROM子句，WHERE子句组成的查询块。
数据控制语言DCL：用来授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，对数据库实行监视等。