Hubble数据库操作手册

要求：
    1.熟悉hadoop生态，了解并使用hdfs、zookeeper、hive。
    2.了解sql语法。
    3.熟练使用常用的linux命令。

1.Hubble简介

1.1 Hubble是什么?

Hubble是一款支持HTAP操作的分布式数据库。

#什么叫 HTAP
    混合事务/分析处理（HTAP Hybrid Transactional/Analytical Processing）在保留原有在线交易功能的同时，也强调了数据库原生计算分析的能力。支持混合负载的数据库能够避免在传统架构中，在线与离线数据库之间大量的数据交互，同时也能够针对最新的业务数据进行实时统计分析。

    在当前新的业务需求下，数据库除了应对操作型业务，还会在业务实时数据监控，数据报告和决策辅助方面有许多数据实时应用的场景。传统架构中OLTP和OLAP两类业务是完全分离的。两者的隔离导致整个系统在数据一致性，数据平台管理上带来了巨大的阻碍。此前，RDBMS和大数据（Hadoop）分别成为数据处理的两个方面，一旦一种架构选择了一种场景，就不得不放弃另一种场景。
    因此，HTAP混合事务/分析处理模式成为数据库发展的一个新要求。

总结：HTAP大白话理解：如何在OLTP单一数据系统上，提供OLAP操作。

1.2 Hubble能干什么?

1.2.1 Hubble能支持HTAP场景

    HTAP数据处理大致可以分成两大类：联机事务处理OLTP（on-line transaction processing）、联机分析处理OLAP（On-Line Analytical Processing）。OLTP是传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。OLAP是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

1.2.2 OLTP操作

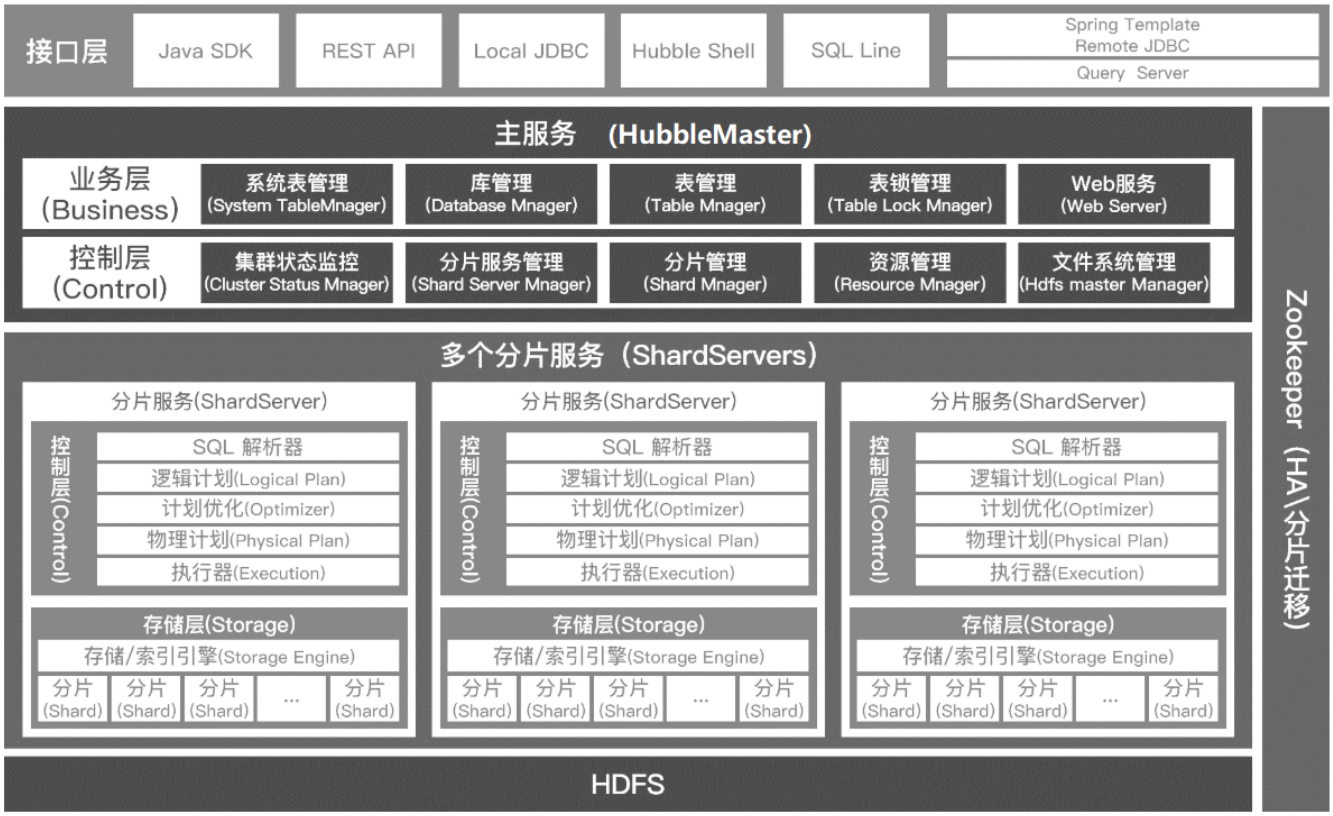
    OLTP，也叫联机事务处理（Online Transaction Processing），表示事务性非常高的系统，一般都是高可用的在线系统，以小的事务以及小的查询为主，评估其系统的时候，一般看其每秒执行的Transaction以及Execute SQL的数量。在这样的系统中，单个数据库每秒处理的Transaction往往超过几百个，或者是几千个，Select 语句的执行量每秒几千甚至几万个。典型的OLTP系统有电子商务系统、银行、证券等，如mysql数据库，就是很典型的OLTP数据库。

1.2.3 OLAP操作

OLAP，也叫联机分析处理（Online Analytical Processing）系统，有的时候也叫DSS决策支持系统，就是我们说的数据仓库。在这样的系统中，语句的执行量不是考核标准，因为一条语句的执行时间可能会非常长，读取的数据也非常多。所以，在这样的系统中，考核的标准往往是磁盘子系统的吞吐量（带宽），如能达到多少MB/s的流量。
常用的AP标准：TPC-DS、TPC-H

1.3 Hubble如何去做?

1.3.1 Hubble的架构图



Hubble数据库主服务进程角色：

```
HubbleMaster:管理节点

ShardServer：数据节点
```

1.3.2 Hubble 产品特性

1.3.2.1 支持海量/高并发

海量/高并发支持

名词解释：即席查*(Ad Hoc)是用户根据自己的需求，灵活的选择查询条件，系统能够根据用户的选择生成相应的统计报表。

1.3.3.2 标准SQL全面的支持

表 1　TPC-DS SQL 查询特征

SQL 特征	查询数量
子表达式	31
关联的子查询	15
不相互关联的子查询	76
Group By	78
Order By	64
Rollup	9
Partition	11
Exists	5
Union	17
Intersect	2
Minus	1
Case	24
Having	5

通过TPC-H和TPC-DS标准测试

表 2 TPC-H 与 TPC-DS

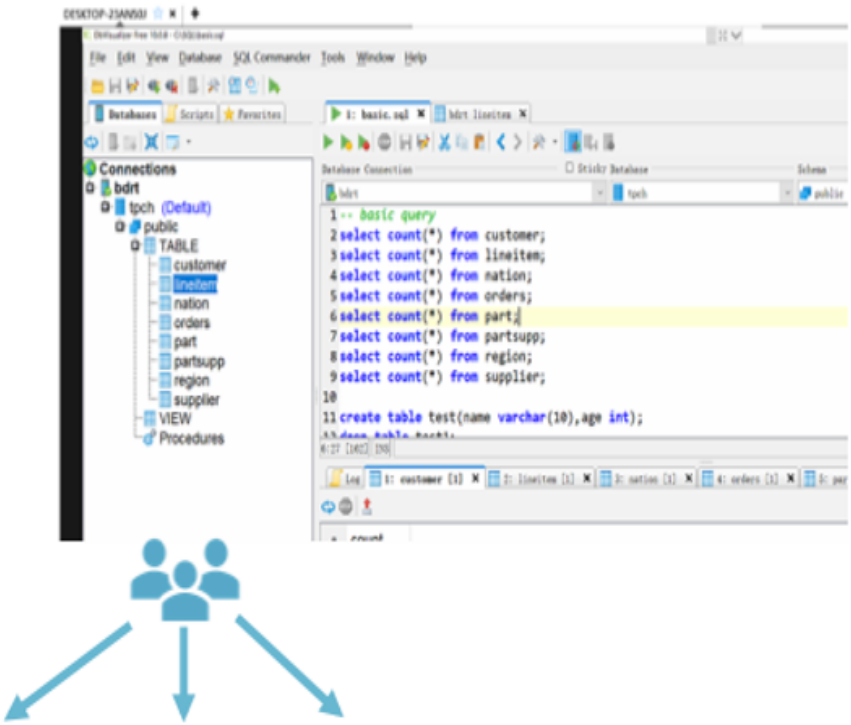
	TPC-H	TPC-DS
数据模型	关系模型第三范式	关系模型、星型模型、雪花模型
SQL 标准	SQL92	SQL99、2003、OLAP
工作负载	8 张表，22 个 SQL	24 张表，99 个 SQL
性能测试	严重依赖于索引，容易被 hack	键控性好，能够比较客观反映系统的真实性能

传统场景无缝迁移

降低参与大数据开发的门槛

1.3.3.3 灵活的查询方式

- 全面SQL支持（DDL,DML,DCL,DQL）
- 支持复杂查询语句，例如：count/group by等
- 封装多种查询调用接口，方便系统对接查询应用



2.Hubble数据库安装部署

2.1 安装准备

2.1.1 Hubble安装环境要求

安装hubble前，首先必须对整个集群环境进行规划，规划包括集群节点、网络、软件等。集群环境必须保证满足建设的集群满足Hubble的基本要求。这些要求包括，网络、软件、硬件要求，下面章节将对这些要求进行详细介绍，列出安装前的硬件、软件基本要求，在满足这些要求的基础上，才可以进行hubble的安装。

2.1.2 硬件环境要求

配置	最低要求	推荐配置
CPU	2路主频2.4GHz每路16核	2路主频2.4GHz以上，每路16核以上
内存	128G	256G及以上
磁盘	6块STAT，单盘容量1T 7200转	10块STAT及以上，单盘2T 7200转（或SSD）
网络	2块千兆网卡	2块万兆网卡
台数	6台	8+

2.1.3 操作系统要求

操作系统	版本
RedHat	
CentOS	7

2.1.4 软件环境要求

软件	版本号
java	jdk1.8以上版本
Hadoop	推荐2.6以上版本
zookeeper	3.4.5
Hive	Hive 2.1.0

2.2 环境检查

2.2.1 系统磁盘分区检查

系统安装和运行需要占用硬盘空间，Hubble需要选择容量超过200GB的磁盘作为数据盘， 对磁盘进行分区时需要遵守以下几点要求：

- 至少要分出swap和加载于`/`的系统分区。
- 推荐系统分区大小为200GB~500GB，并将该分区挂载到`/`目录。
- 推荐把每个物理磁盘挂载在/mnt/disknn (nn为1至2位的数字) 上不同的挂载点。建议使用XFS文件系统。使用这些目录在作为HDFS数据目录及Hubble中间结果溢出时使用的目录。
- Hubble的溢出目录不能放在系统分区，以避免空间不足和IO竞争。同时也建议不要将HDFS数据分区和系统分区放在同一块磁盘上以避免IO竞争。
- 请在某挂载盘上为Hubble预留不小于200GB的空间，并将Hubble的spill dir设置为该挂载盘的某个目录（例如/mnt/disk1/hubbledata/）。

2.2.2 网络检查

- 安装hubble需要最低的网络是千兆以太网，确认网络带宽，否则网络会成为影响性能的瓶颈。
- 确认集群的节点数量并确认每个节点的主机名及IP地址。
- 确认集群的管理节点。
- 确认集群互信正常。
- 确认集群hadoop及hive、zk正常运行。

2.2.3 时钟同步

确认如何进行时间同步。确认管理节点将负责所有服务器上的时间的同步，并且需要决定是否使用外部的NTP服务。如果不使用外部NTP服务，集群中所有服务器的时间是相同的，但这个时间有可能不是标准时间，这有可能导致集群与外部连接时产生错误。

2.2.4 其他系统参数检查

- 禁掉SELinux和iptables
- 系统最大打开文件数优化
- 磁盘调优，因为hadoop系统的原因，不需要记录文件时间，所以在配置`/etc/fstab`中进行如下优化：

```
 #
 # /etc/fstab
 # Created by anaconda on Thu Jan 25 09:57:56 2018
 #
 # Accessible filesystems, by reference, are maintained under '/dev/disk'
 # See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
 #
    UUID=655f9adf-71bc-4b96-b345-7bb88feba3d0 /                       xfs     defaults
       0 0
    UUID=9A28-E21A          /boot                   vfat    umask=0077,shortname=winnt
0 0
    UUID=28d8581f-dabf-49af-aa48-51000768dc08 /data                   xfs     defaults
       0 0
    UUID=70ed941a-780d-4f3c-8441-a7a8fa30c762 /opt                    xfs     defaults
       0 0
    UUID=91cd4682-1c13-422e-ac3a-1088f35d5b07 swap                    swap    defaults
       0 0
    UUID="fdfeade5-13f7-4083-9089-e7b7924dc43b" /data1                xfs
defaults,noatime,nodiratime        0 0
    UUID="2cd97464-c329-4480-bf31-b9a3b9523f3c" /data2                xfs
defaults,noatime,nodiratime        0 0
```

2.3 安装Hubble数据库

2.3.1 解压及注册license

2.3.1.1 解压安装包

将hubble的安装包解压，目前最新稳定版本的hubble是3.4.0，hubble的安装包是hubble-3.4.0.tar.gz。

```
#上传hubble tar文件到安装目录，并完成解压操作
tar -zxvf hubble-3.4.0.tar.gz -C .
#修改安装包名称
mv hubble-3.4.0 hubble
```

默认使用hadoop相关的权限用户，如果是使用root用户，需要赋权给此目录,如hadoop的用户为`hadoop`。

```
#hubble解压包下用户组全部更改hadoop
chown -R hadoop:hadoop hubble

备注：需要在linux下创建hadoop用户，组
```

2.3.1.2 注册license

hubble-license脚本

```
bin/hubble-license.sh 命令提示如下
Usage: hubble-license.sh [options] [...]
 -i,--input  input License code
 -m,--machineInfo get current machine code
 -s,--status get current License status
 -u,--update  update License code,can be string,can be text
```

配置license的步骤

```
***注意：注册license只需要在master节点上操作即可，待修改完hubble的conf目录配置文件，分发安装包即可。
# 1.输出机器码
bin/hubble-license.sh -m
注:以上命令会输出机器码
例:
请凭借机器码像管理人员索取License文件，您的机器码为:
140d3742faffc204b9743f133f79e667d8dcd3de3e9db51a0f5b92634f8bd0e63cf7217a2fd67e9d7f9503d5cd8b
5590a73ae962ceabbaa3e6f59c6e9a5db82a04b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7
250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b3
3fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f4
04b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb72503
75f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7
250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b3
3fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f4
04b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb72503
75f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7
250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b3
3fb7250375f40xB4B0

# 2.根据机器码联系相关负责人（王伯阳，郑习兵），得到license文件
需要提供的信息：项目编号、节点个数、生成的license文件、授权时长
相关负责人会反馈一个授权的license文件
例:
140d3742faffc204b9743f133f79e667d8dcd3de3e9db51a0f5b92634f8bd0e63cf7217a2fd67e9d7f9503d5cd8b
5590a73ae962ceabbaa3eb1ee8624bbcfd1b48487159d6f4d9cabe52a7967ea0bdaf11705955542eb5ffe300690e
e494f83b0256b460dcd950cec2f73de1564c29296bca89f5d73fb4edb3894ec1818bacc25d8e832d9c0cc18f04b3
3fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f4
04b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb72503
75f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7
250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b3
3fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f4
04b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb72503
75f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7
250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b33fb7250375f404b3
3fb7250375f40x1ACE

# 3.注册license
bin/hubble-license.sh -i <license文件>
注：这里参数后面跟的是license的文件，不是license的字符串

#4.更新license
bin/hubble-license.sh -u <新license文件>
注：这里参数后面跟的是license的文件，不是license的字符串
```

```
#5. 检查license的状态
bin/hubble-license.sh -s
```

2.3.2 配置文件介绍

hubble数据库的`conf`目录。

```
core-site.xml         复制hadoop的core-site.xml文件到此目录下。
hdfs-site.xml         复制hadoop的hdfs-site.xml文件到此目录下。
hplsql-site.xml       hubble的存储过程配置文件（目前是实验性质的）
hubble-default.xml    hubble的默认配置，不需要更改。需要更改时在hubble-site.xml中修改，会覆盖default中
的默认配置。
hubble-env.sh         hubble的环境变量设置文件。
hubble-site.xml       hubble的主要配置参数文件。
log.properties        hubble的日志配置文件。
shardservers          hubble的shardserver节点信息。
catalog               目录下包含hubble连接其他数据源时需要的信息。
```

2.3.3 配置参数介绍

2.3.3.1 hubble-env.sh

```
#!/bin/bash

#配置jdk运行环境
export JAVA_HOME=/opt/jdk1.8.0_191
#配置hadoop环境
export HADOOP_HOME=/opt/hadoop
#配置hubble数据库安装目录
export HUBBLE_HOME="/opt/hubble"

export HUBBLE_DATA_DIR=${HUBBLE_HOME}/data

#JVM参数
HUBBLE_OPTS="-server"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+UseG1GC"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:G1HeapRegionSize=32M"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+UseGCOverheadLimit"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+ExplicitGCInvokesConcurrent"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+HeapDumpOnOutOfMemoryError"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+ExitOnOutOfMemoryError"
HUBBLE_OPTS="$HUBBLE_OPTS -Dlog.levels-file=$HUBBLE_HOME/conf/log.properties"

#配置master节点JVM参数（推荐数值为当前节点内存的60%以内，以120GB内存为例）
export HUBBLE_MASTER_OPTS="$HUBBLE_OPTS -Xss256k -Xms60g -Xmx60g"

#配置shardserver节点JVM参数（推荐数值为当前节点内存的60%以内，以120GB内存为例）
export HUBBLE_SHARDSERVER_OPTS="$HUBBLE_OPTS -Xss256k -Xms60g -Xmx60g"

#hubble client配置（使用默认配置即可）
export HUBBLE_CLIENT_SQL_OPTS="$HUBBLE_OPTS -Xss256k -Xms1g -Xmx2g"
```

2.3.3.2 hubble-site.xml

```
<configuration>
    <property>
                <name>hubble.zookeeper.quorum</name>
                <value>master:2181</value>
                <description>zookeeper集群的地址</description>
        </property>
        <property>
                <name>hubble.rootdir</name>
                <value>hdfs://bdpha/hubble</value>
                <description>hubble数据库在hdfs集群的存储路径</description>
        </property>
        <property>
                <name>hubble.master.address</name>
                <value>master</value>
                <description>master节点的主机名</description>
        </property>
        <property>
                <name>hubble.master.port</name>
                <value>31006</value>
                <description>hubble master节点RPC端口</description>
        </property>
        <property>
                <name>hubble.shard.server.port</name>
                <value>31010</value>
                <description>hubble shardserver节点RPC端口</description>
        </property>
        <property>
                <name>zookeeper.znode.parent</name>
                <value>/hubble</value>
                <description>hubble数据库注册zookeeper的路径</description>
        </property>
        .
        .
        .
</configuration>
```

2.3.3.3 shardservers

```
#shardserver的主机名，不含master节点主机名
node1
node2
node3
node4
node5
node6
```

2.3.3.4 catalog目录

2.3.3.4.1 简介:

在hubble中，catalog相当于给数据源一个命名，如配置了三个数据源，一个hubble源，一个hive源，一个mysql源，即数据分别存储在hubble、hive、mysql中，这样，在hubble中就存在了三个数据源，此时hubble里面有三个catalog，分别是：hive.properties,mysql.properties（hubble源默认配置）。
目录路径：~/hubble/conf/catalog/xxx.properties

2.3.3.4.2 数据源适用场景

#hubble数据源适用场景
hubble源在hubble中默认启用，不需要调整catalog配置，hubble源底层存储在HDFS上。适用于大量写入、少量更新、随机读写即OLTP的场景。

#hive数据源适用场景
hive源在hubble中使用时并没有使用hive的查询引擎以及hive的环境变量等，只是使用了hive底层存储的数据以及元数据服务。访问元数据服务主要来获取数据结构。使用数据为了与hive兼容。hive源适用于OLAP场景。
hive支持的文件数据格式：ORC，Parquet，Avro，RCFile，SequenceFile，JSON，Text

2.3.3.4.3 具体配置信息

配置hive数据源

```
#配置hive源，修改hive.properties文件
##配置数据源
catalog.name=hive
##thrift接口
hive.metastore.uri=thrift://master:9083
hive.config.resources=/opt/hadoop/etc/hadoop/core-site.xml,/opt/hadoop/etc/hadoop/hdfs-site.xml
hive.allow-add-column=true
hive.allow-drop-column=true
hive.allow-drop-table=true
hive.allow-comment-table=true
hive.allow-rename-table=true
hive.allow-rename-column=true
hive.writer-sort-buffer-size=2GB
```

配置mysql数据源

```
catalog.name=mysql
connection-url=jdbc:mysql://[serverName[\instanceName][:portNumber]]
connection-user=root
connection-password=secret
```

注意：

对于关系型数据库，只是作为多源的一部分来处理的，只实现了简单的类型转换，如bigint,smallint,int,double,char(n),varchar(n),date这些，其他复杂的类型可能不支持。

配置多个hive数据源

```
配置多个hive数据源，可以在catalogs下面建多个hive.properties文件，例
hive192.properties
    catalog.name=hive
    ##thrift接口
    hive.metastore.uri=thrift://cluster192:9083

hive151.properties
    catalog.name=hive
    ##thrift接口
    hive.metastore.uri=thrift://cluster153:9083

hive109.properties
    catalog.name=hive
    ##thrift接口
    hive.metastore.uri=thrift://cluster109:9083
```

2.3.3.5 hubble-default.xml

```
<configuration>
        <property>
                <name>node.environment</name>
                <value>production</value>
                <description>当前环境类型</description>
        </property>
        <property>
                <name>http-server.http.port</name>
                <value>31009</value>
                <description>jdbc.cli.web服务端口号</description>
        </property>
        <property>
                <name>discovery-server.enabled</name>
                <value>true</value>
                <description>是否自动发现服务</description>
        </property>
</configuration>
```

2.3.3.6 hdfs-site.xml、core-site.xml

```
直接将hdfs集群对应的配置文件拷贝过来即可
```

2.3.4 启动集群

2.3.4.1 分发

因为hubble是配置统一的，所以只需要在主节点上配置完后分发即可，不需要每个节点单独配置。

```
scp -r hubble <节点>:<目录>
```

2.3.4.2 启停

启动需要在`hubble`根目录下执行：

```
bin/start-hubble.sh
```

停止：

```
bin/stop-hubble.sh
```

额外包含一些其他脚本，与hadoop脚本类似。

`hubble-daemon.sh`

单独启动master或者某一个节点的shardserver,进入需要启动的节点，到bin目录下执行：

```
./hubble-daemon.sh start master #对应master启动
```

```
./hubble-daemon.sh start shardserver #对应shardserver启动。
```

单独停止master或者某一个节点的shardserver,进入需要启动的节点，到bin目录下执行：

```
./hubble-daemon.sh stop master
```

```
./hubble-daemon.sh stop shardserver
```

`hubble-daemons.sh`

批量命令，如批量停止shardserver。

在主节点bin目录下执行：

```
./hubble-daemons.sh stop shardserver
```

2.4 安装Hubble客户端工具

2.4.1 JDK环境安装

```
# jdk安装版本
jdk1.8以上版本

# jdk-win64位下载链接
    https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

# 验证java环境是否安装成功
    1.win+r,输入cmd，回车
    2.cmd命令框中输入
        java -version
```

```
3.打印输出，安装成功
    C:\Users\wulong>java -version
    java version "1.8.0_181"
    Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
    Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

2.4.2 Dbeaver客户端工具安装

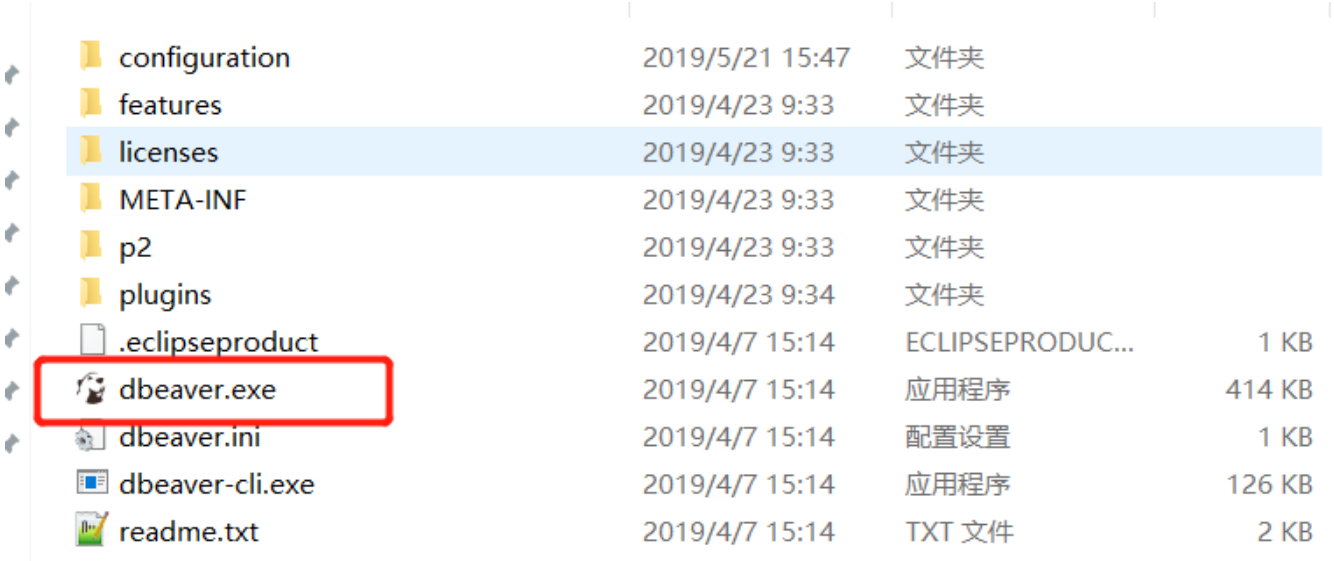
2.4.2.1 hubble客户端工具安装

1.下载dbeaver客户端工具（64位机），链接地址

```
https://dbeaver.io/download/
win64位下载压缩包名称：dbeaver-ce-6.0.2-win32.win32.x86_64.zip
解压到本地：dbeaver目录下
```

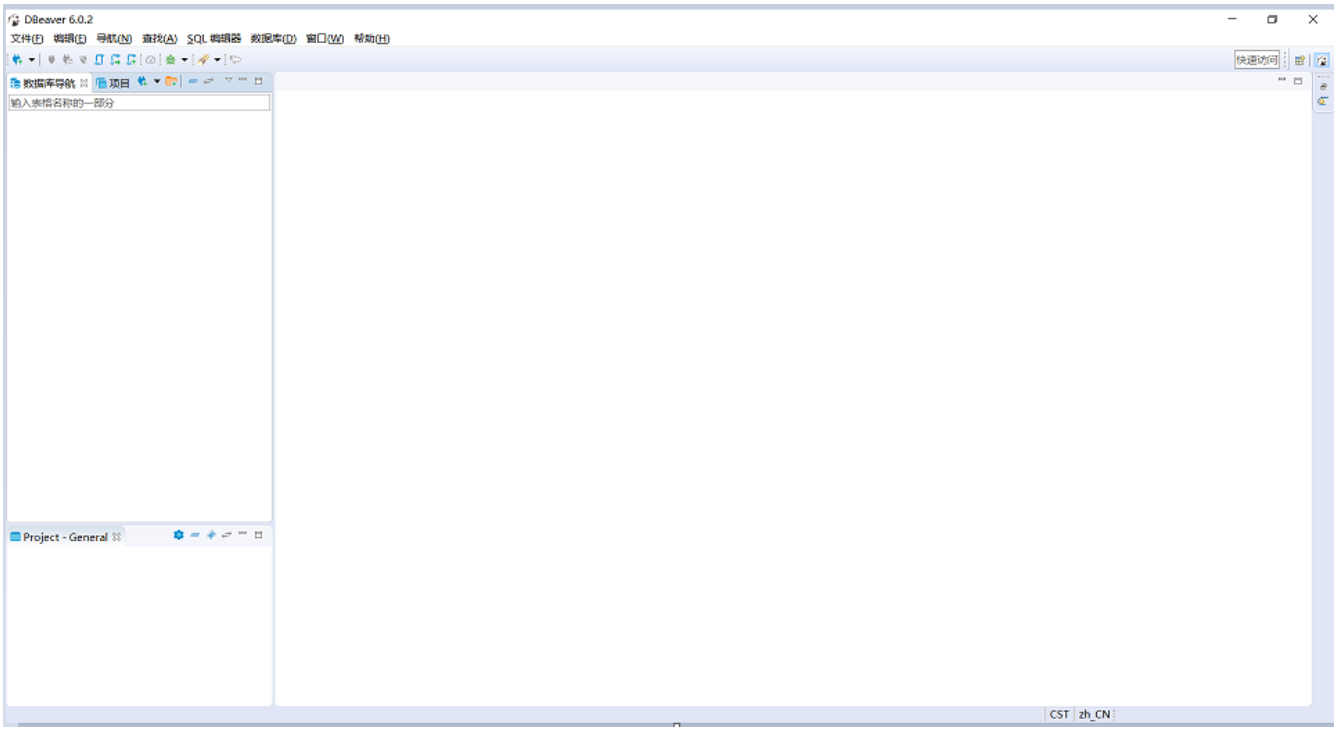
2.启动dbeaver客户端

解压文件目录：



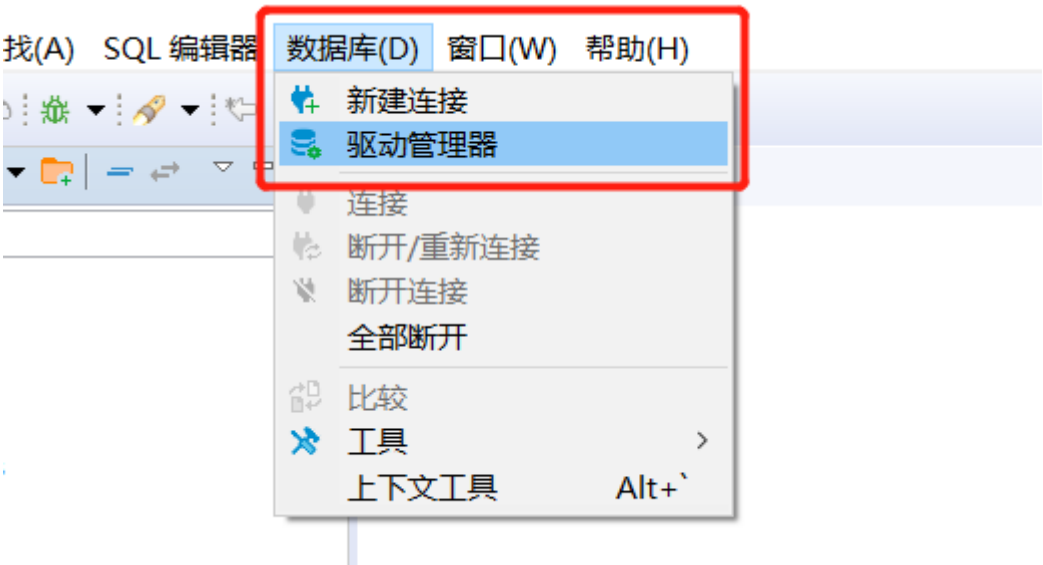

```
增加dbeaver的jvm内存：打开dbeaver.ini文件，增加 -Xms -Xmx两个参数
```

3.打开操作界面

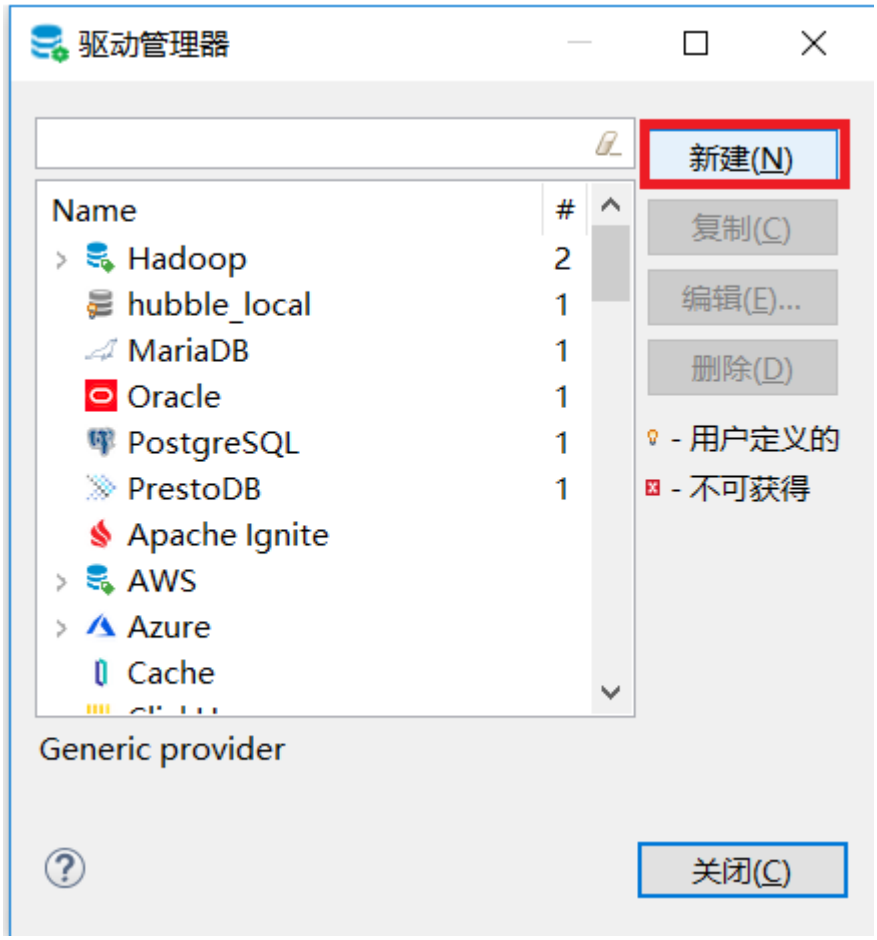


2.4.2.2 hubble客户端工具配置

1.点击数据库->驱动管理器

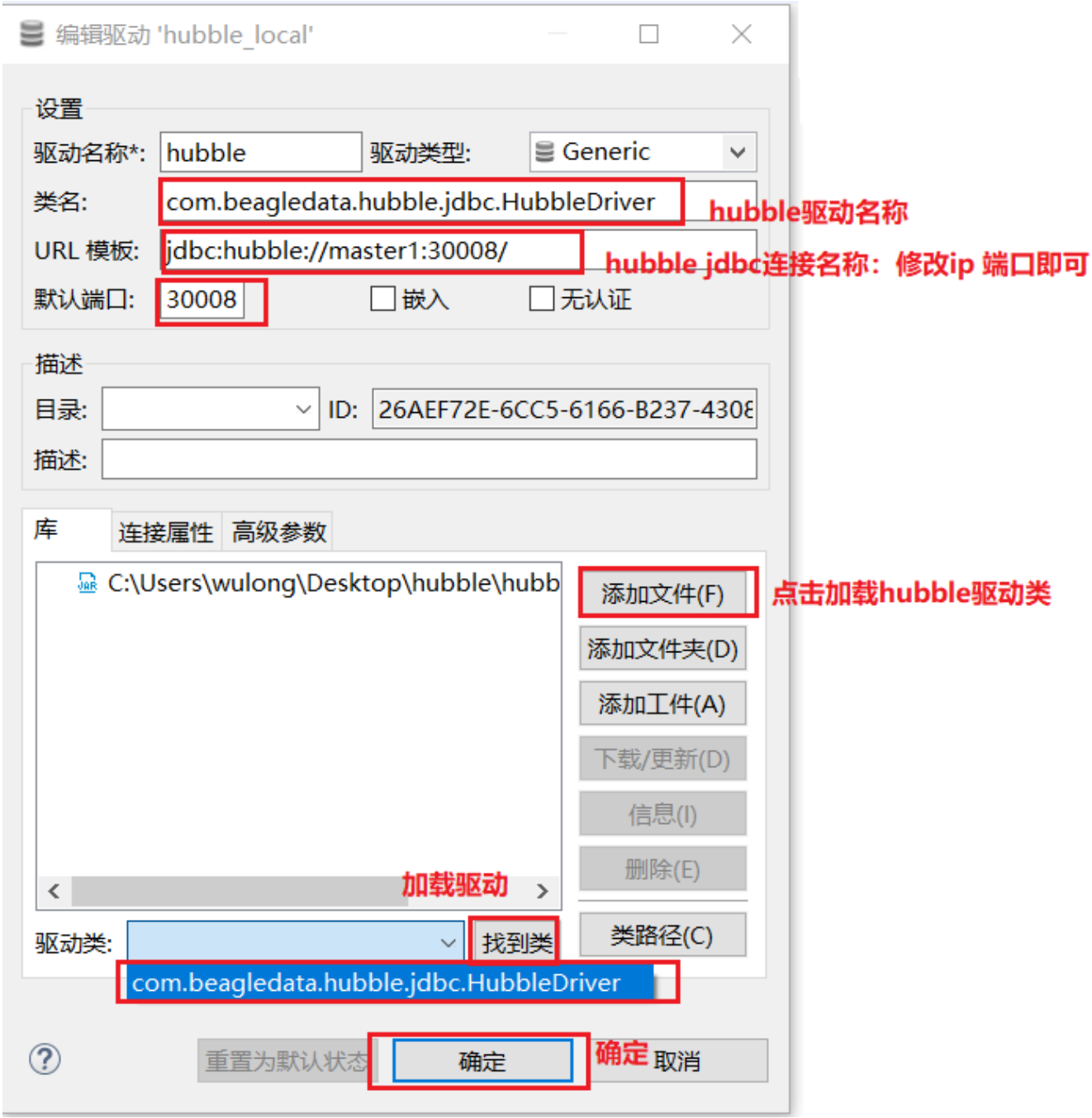


2.点击新建



3.输入配置信息

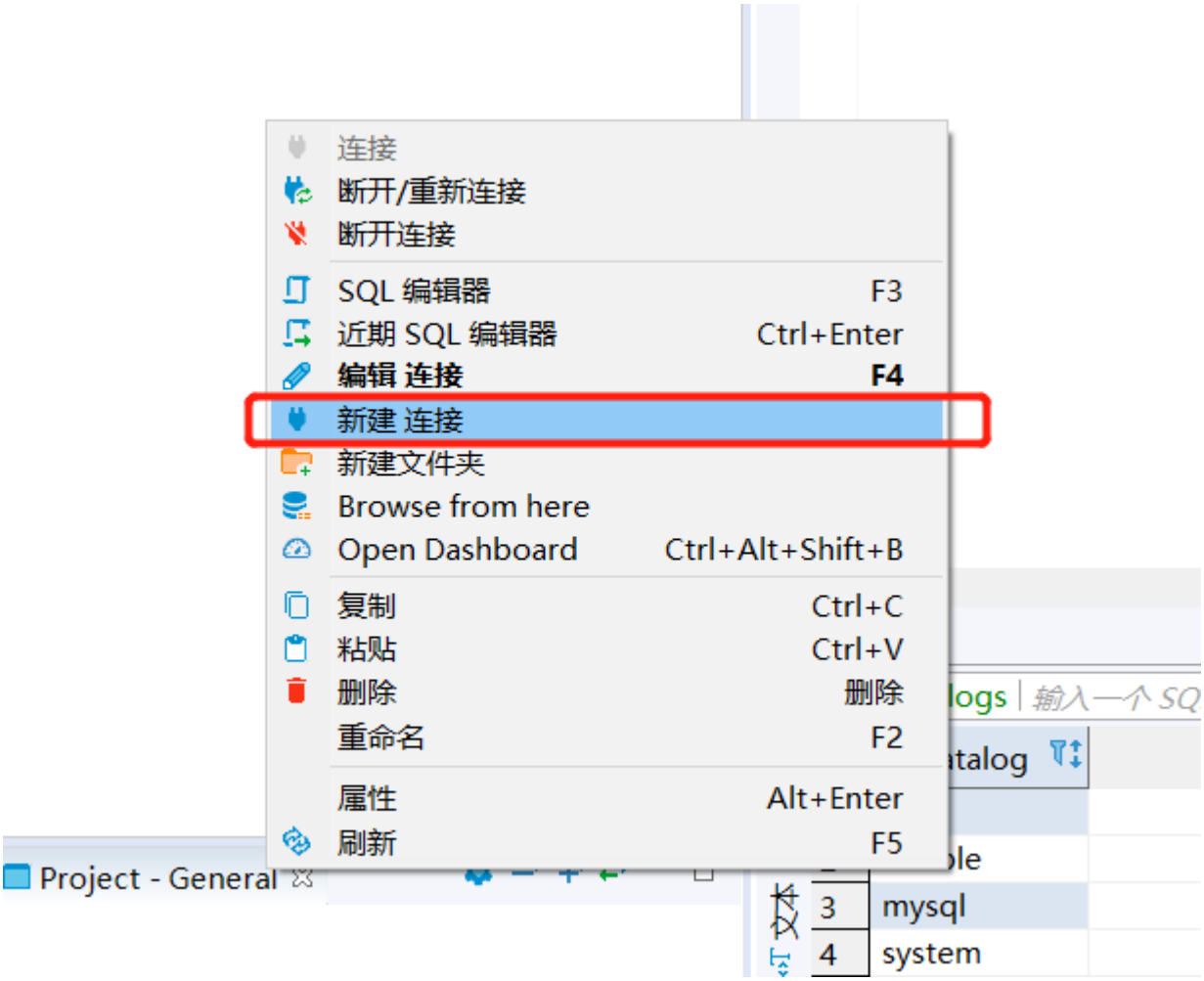
```
hubble数据库驱动类名:  com.beagledata.hubble.jdbc.HubbleDriver
jdbc 链接:  jdbc:hubble://127.0.0.1:30008/
```



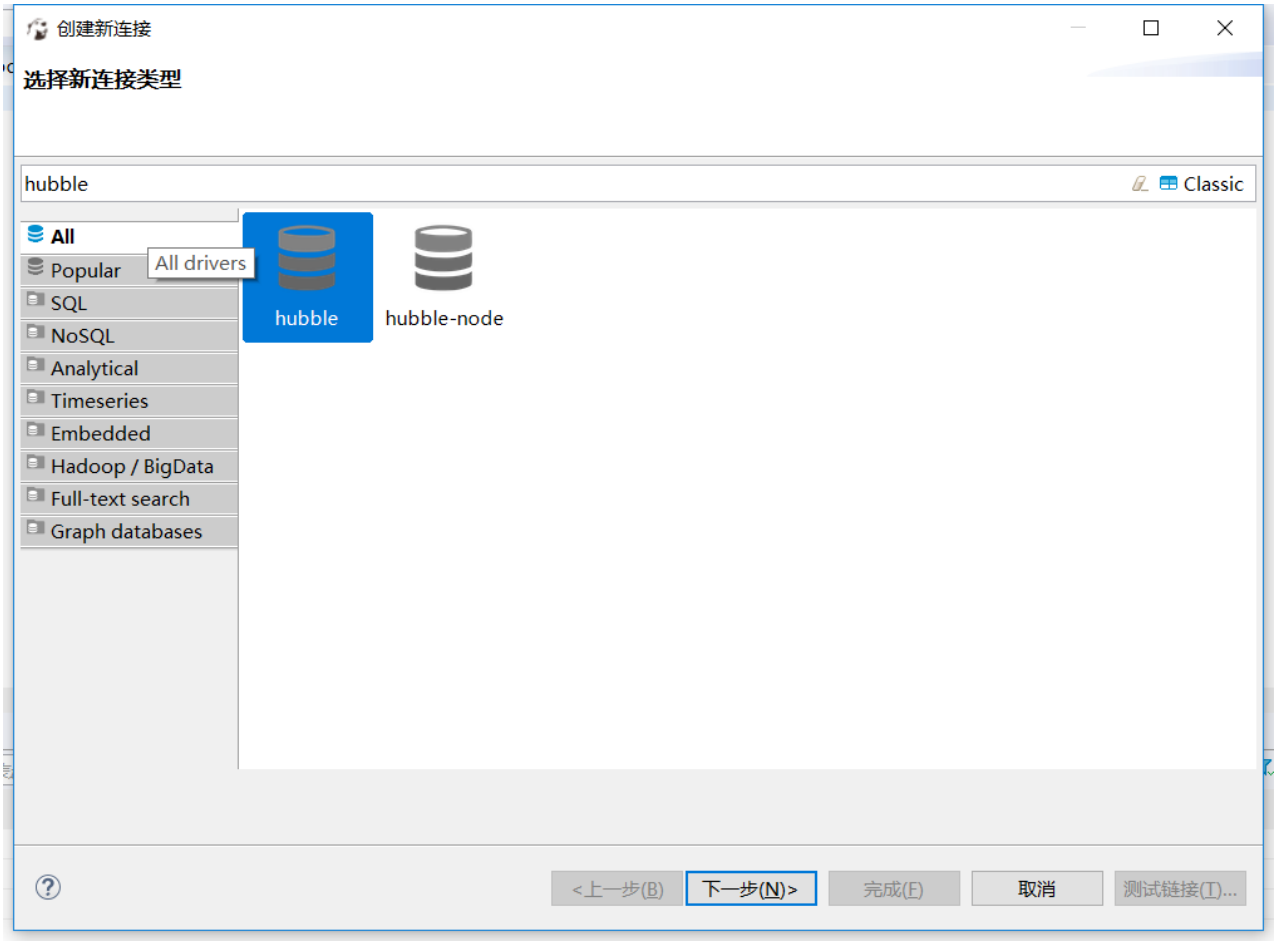
4.点击确定

2.4.2.3 连接配置hubble数据库

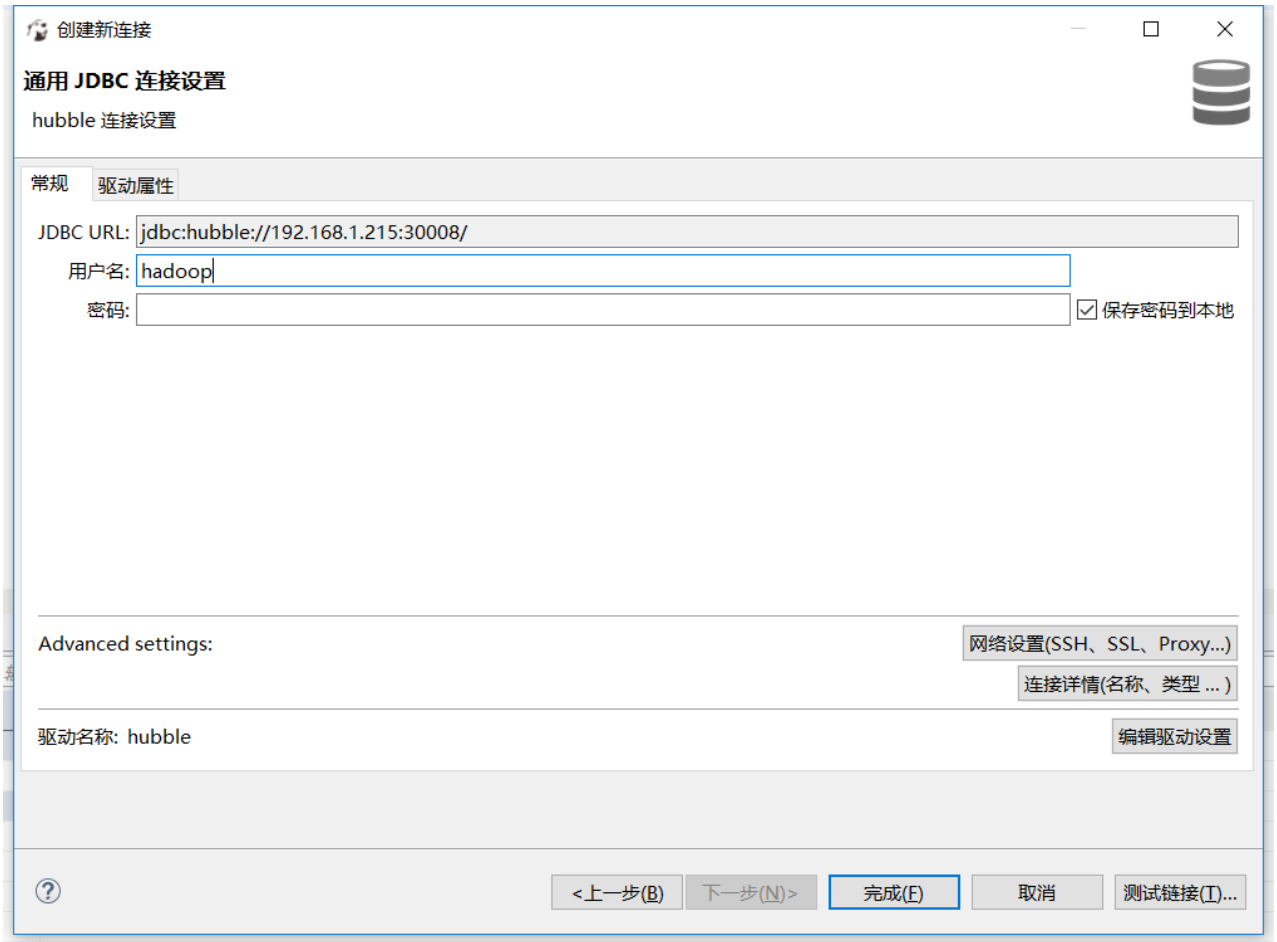
1. 选择新建连接



2. 选择hubble数据库连接

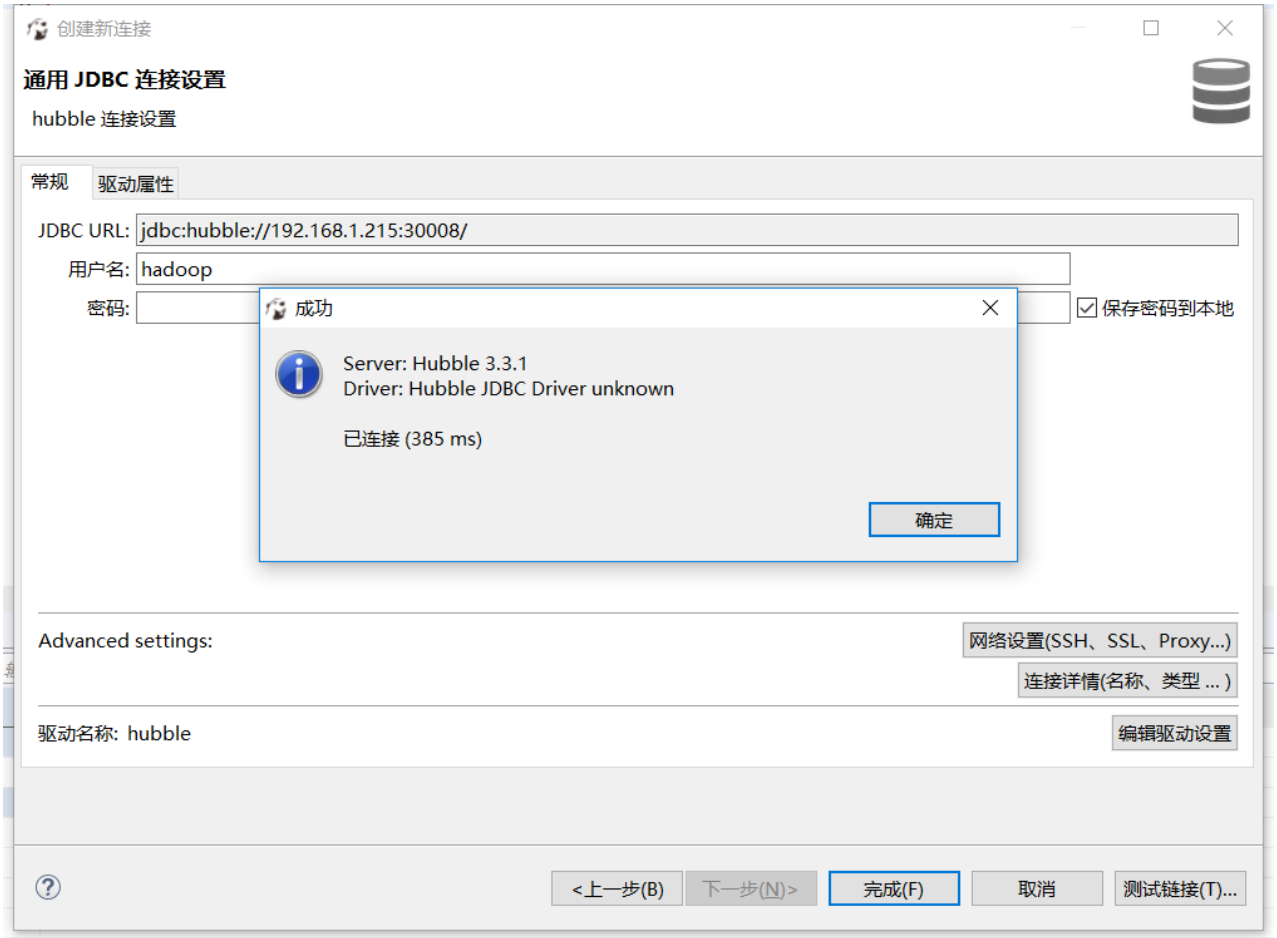


3. 点击下一步，输入用户信息

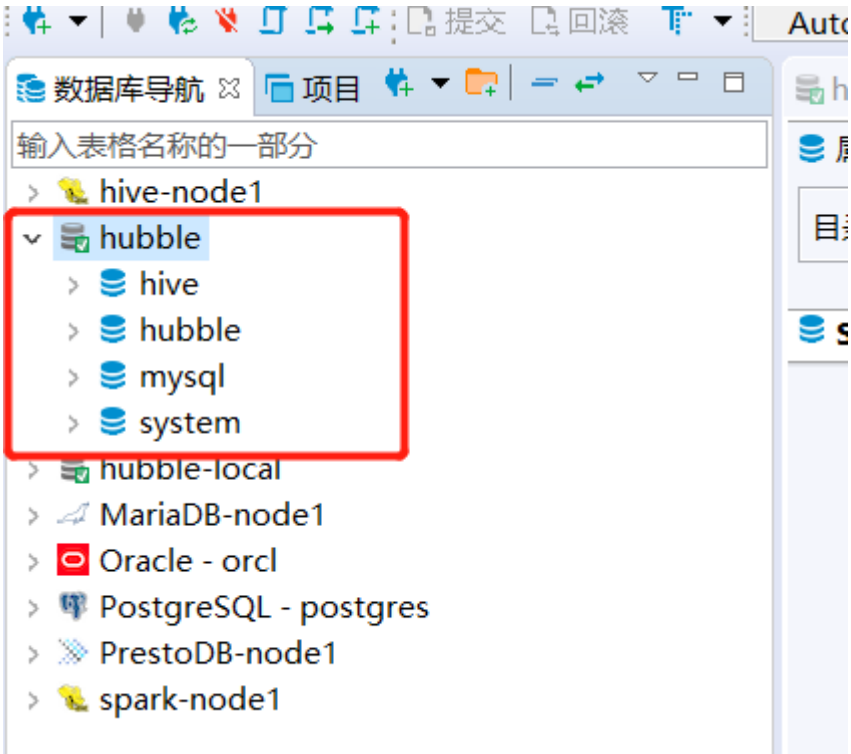


注意：如果不需要权限认证，密码不填

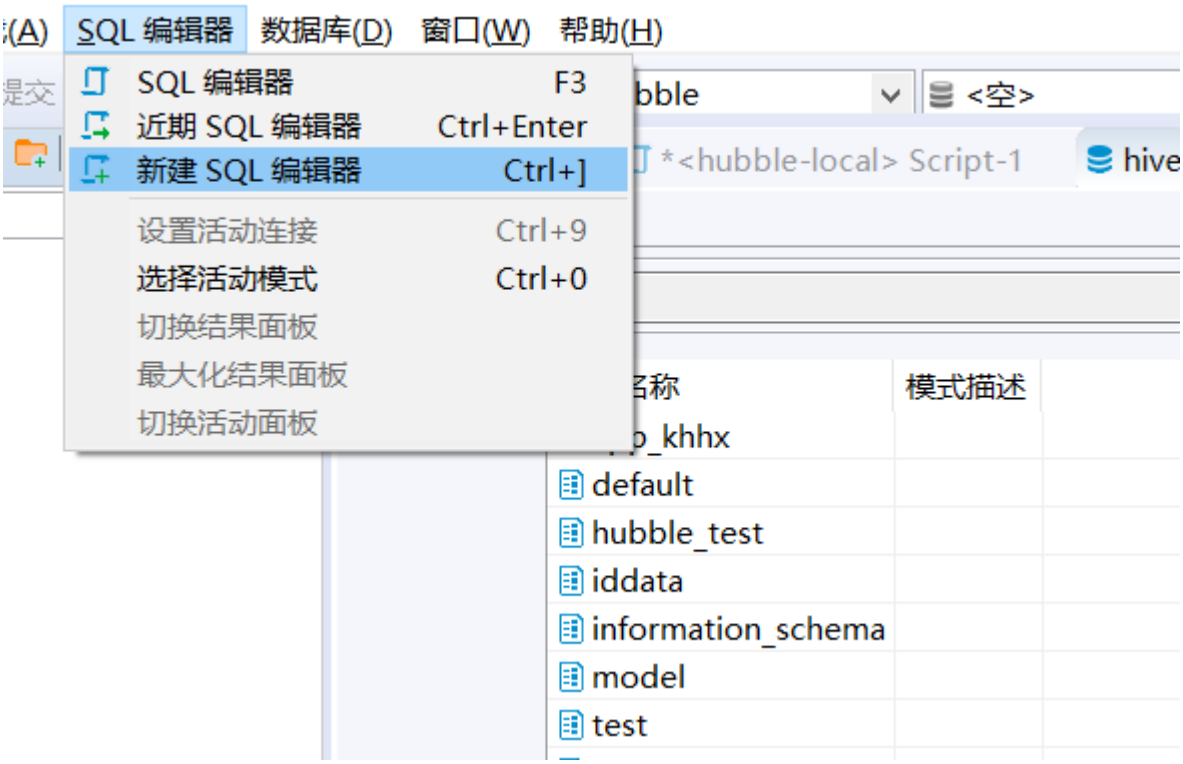
4. 点击测试链接，连接成功

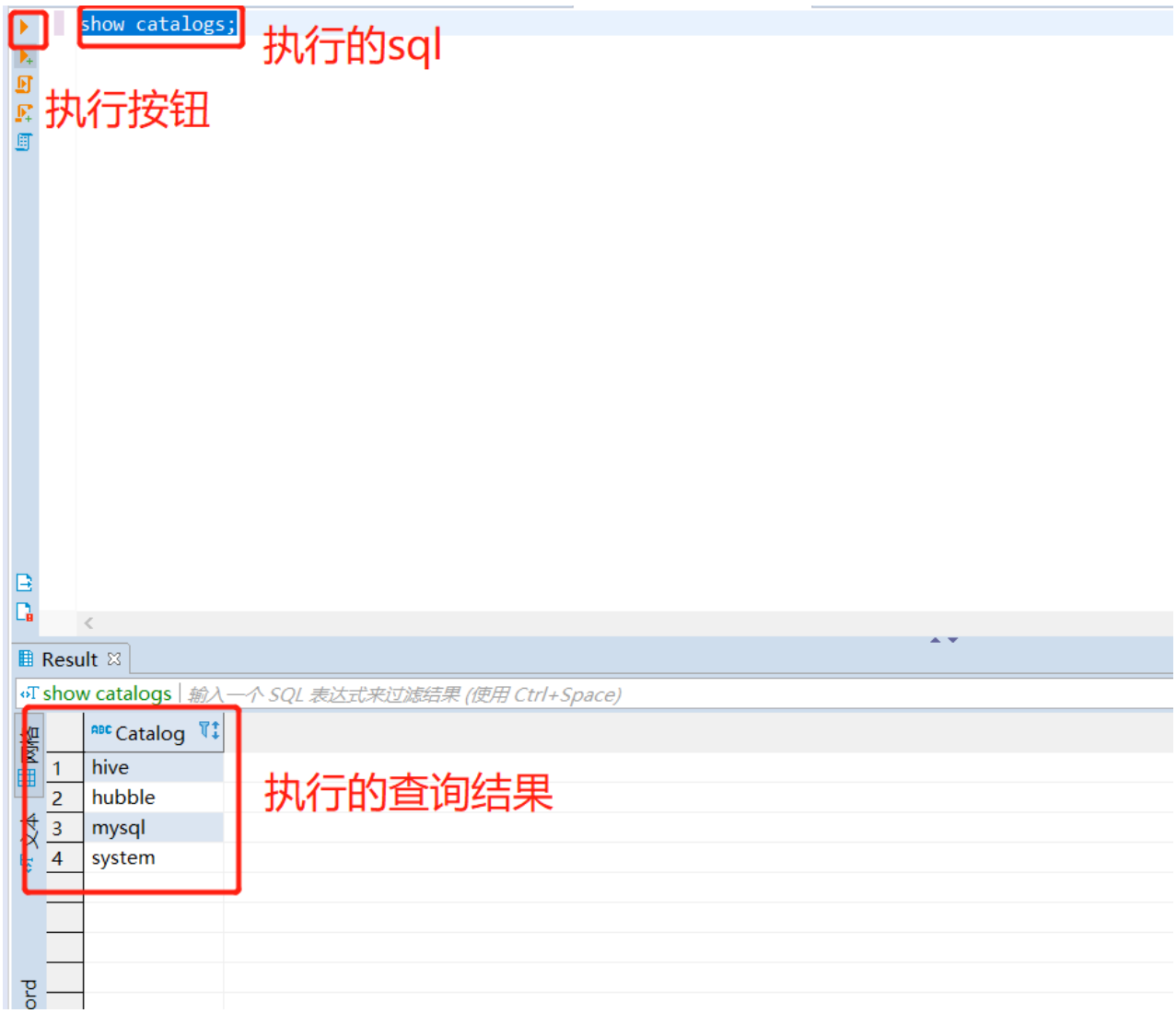


5. 双击建立的连接，操作数据库



6. 打开sql编辑器,执行测试sql


(A)
SQL 编辑器
数据库(D)
窗口(W)
帮助(H)
SQL 编辑器 F3
近期 SQL 编辑器 Ctrl+Enter
新建 SQL 编辑器 Ctrl+]
设置活动连接 Ctrl+9
选择活动模式 Ctrl+0
切换结果面板
最大化结果面板
切换活动面板
<空>
*<hubble-local> Script-1
hive
模式描述
default
hubble_test
iddata
information_schema
model
test


show catalogs;
执行的sql
执行按钮
Result
show catalogs
输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)
Catalog
1 hive
2 hubble
3 mysql
4 system
执行的查询结果

7. 完成安装

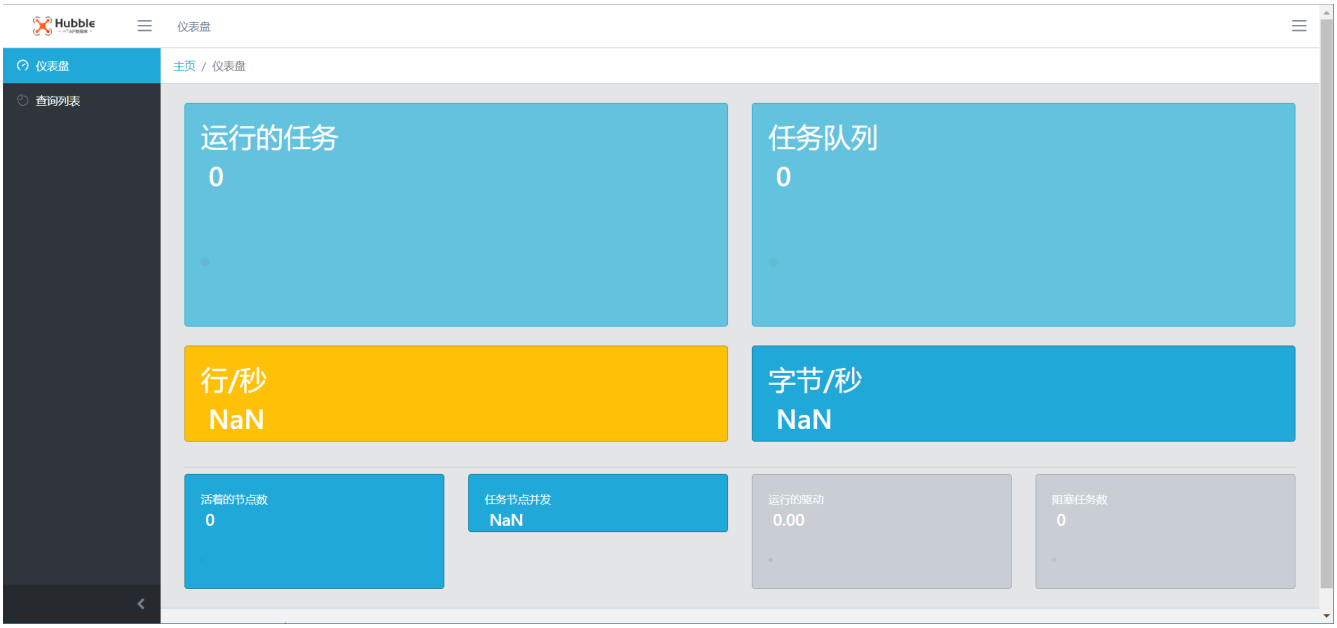
2.7 Hubble数据库管理页面

2.7.1 访问端口

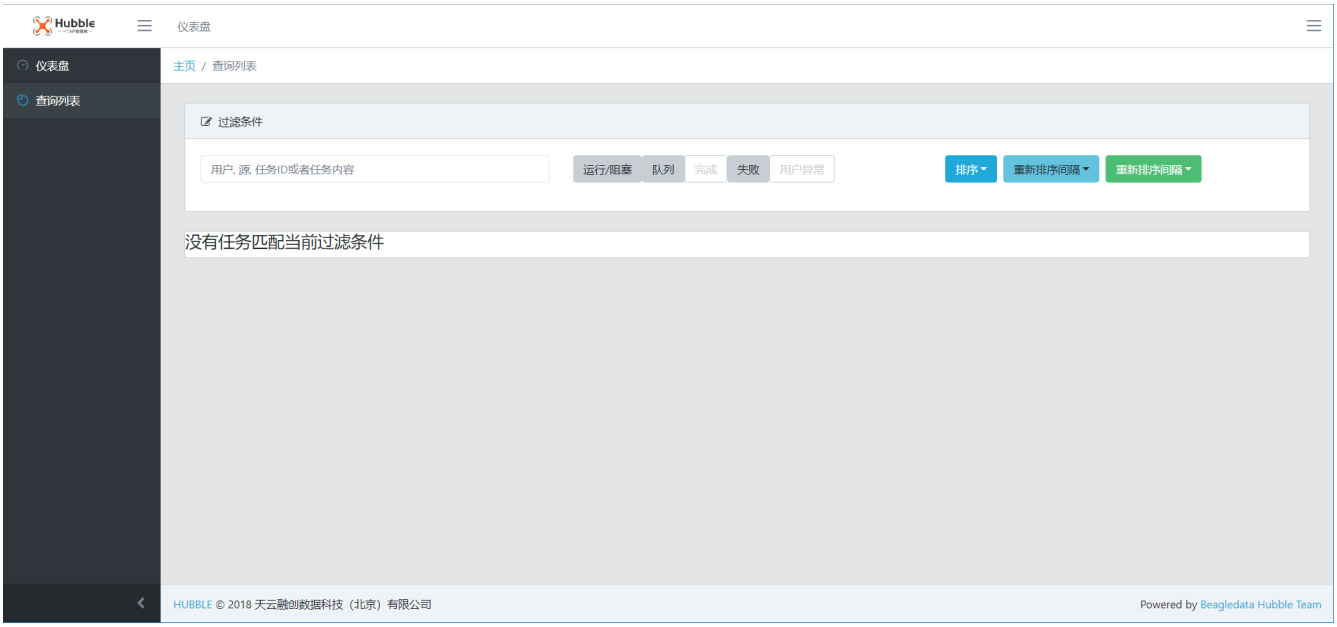
```
#hubble-default.xml配置对外访问变量
        <property>
                <name>http-server.http.port</name>
                <value>31009</value>
                <description>jdbc访问端口，web页面访问端口，hubble cli访问端口</description>
        </property>

#Http访问url：
        http://192.168.1.215:31009/
```

2.7.2 集群监控页面



2.7.3 任务监控页面



2.8 Hubble数据库优化

2.8.1 hubble-site.xml的配置

1.hubble.cs.spiller-spill-path

```
<property>
        <name>hubble.cs.spiller-spill-path</name>
        <value>/data1/hubble/spill,/data2/hubble/spill</value>
        <description>写入临时磁盘空间的目录，可以用","分割的多个目录</description>
</property>
```

2.query.max-memory

```
<property>
        <name>query.max-memory</name>
        <value>3000GB</value>
        <description>当前集群查询使用最大内存</description>
</property>
```

3.query.max-memory-per-node

```
<property>
        <name>query.max-memory-per-node</name>
        <value>40GB</value>
        <description>单个节点查询时的最大的内存限制</description>
</property>
```

4.query.max-total-memory-per-node

```
<property>
        <name>query.max-total-memory-per-node</name>
        <value>40GB</value>
        <description>单个节点查询时占用的全部内存，包括用户内存和系统内存</description>
</property>
```

5.http-server.max-request-header-size

```
<property>
        <name>http-server.max-request-header-size</name>
        <value>10MB</value>
        <description>每次http请求消息头的最大长度，请求sql语句传输的最大size</description>
</property>
```

6.exchange.http-client.request-timeout

```
<property>
        <name>exchange.http-client.request-timeout</name>
        <value>160s</value>
</property>
```

2.8.2 hubble-env.sh配置

2.8.2.1 hubble系统运行环境配置

```
#配置hubble数据库，运行时java环境（hubble默认支持jdk 1.8以上版本）
export JAVA_HOME=/opt/jdk1.8.0_191

#配置hadoop环境
export HADOOP_HOME=/opt/hadoop

#配置hubble数据库，安装部署目录
export HUBBLE_HOME="/opt/hubble-deploy"

#Hubble数据库数据目录
export HUBBLE_DATA_DIR=${HUBBLE_HOME}/data
```

2.8.2.2 JVM参数调优

```
#配置GC参数
HUBBLE_OPTS="-server"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+UseG1GC"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:G1HeapRegionSize=32M"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+UseGCOverheadLimit"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+ExplicitGCInvokesConcurrent"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+HeapDumpOnOutOfMemoryError"
HUBBLE_OPTS="$HUBBLE_OPTS -XX:+ExitOnOutOfMemoryError"
```

```
#配置JVM参数
#配置master节点JVM参数，推荐配置xmx、xms数值为节点全部内存大小的一般以上，且数值相同
#以最大128GB内存为例：
export HUBBLE_MASTER_OPTS="$HUBBLE_OPTS -Xss256k -Xms60g -Xmx60g"

#配置shardserver节点JVM参数，推荐配置xmx、xms数值为节点全部内存大小的一般以上，且数值相同
#以最大128GB内存为例：
export HUBBLE_SHARDSERVER_OPTS="$HUBBLE_OPTS -Xss256k -Xms60g -Xmx60g"

#使用此默认配置即可
export HUBBLE_CLIENT_SQL_OPTS="$HUBBLE_OPTS -Xss256k -Xms1g -Xmx2g"
```

2.8.3 hubble-default.xml配置

```
#配置jdbc访问端口，web管理页面端口，hubble cli端访问端口
        <property>
                <name>http-server.http.port</name>
                <value>31009</value>
                <description>jdbc访问端口，web页面访问端口，hubble cli访问端口</description>
        </property>
```

2.8.4 用户临时变量配置

```
#查看用户临时变量
执行sql：show session;

#常用临时用户变量
1.hubble.load_data_module
  参数用法：用户加载数据时，设置导入模式，只对当前用户session生效
    set session hubble.load_data_module=true;
2.task_writer_count
  参数用法：设置写入线程数，
  推荐：writer-count参数的数量与单个节点的cpu核心数相同，但是不要超过单个节点cpu核数的2倍（其数值为2的整数倍）。
    set session task_writer_count=32;（32核服务器推荐配置）
```

3.Hubble数据库的使用

3.1 Hubble数据库基本介绍

3.1.1 Hubble目录层次介绍

3.1.1.1 catalog

在Hubble中，可以使用常见的数据库对象，按照SQL标准，分为了catalog,schema,table。我们可以把它们理解为一个容器或者数据库对象命名空间中的一个层次，主要用来解决命名冲突问题。从概念上说，一个数据库包含多个catalog，每个catalog又包含多个schema,而每个schema又包含多个数据库对象（表、视图、字段等），反过来讲一个数据库对象必然属于一个schema，而该schema又必然属于一个catalog，这样我们就可以得到该数据库对象的完全限定名称从而解决命名冲突问题了。例如一个数据库表的完全限定名称就可以表示为：Catalog名称.Schema名称.表名称。

举个例子，比如想查hubble存储下test库下的test表，可以使用：
查询hubble中的数据：

```
select * from hubble.test.test;
```

查询mysql中的数据：

```
select * from mysql.test.test;
```

查询hive中的数据：

```
select * from hive.test.test;
```

3.1.1.2 database

database（即schema）是存放一组表的目录。hubble catalog自带两个系统库information_schema,hubble。

在所在数据库内可以直接使用对象名指代对象，如果要使用其他数据库中的对象，需要使用 <catalog_name>.<database_name>.<object_name> 来指代，例如 hubble.my_db.my_table。您也可以用 USE 指令切换使用的数据库。创建库也需要指定连接源，例如：

```
create database hubble.my_db;
```

3.1.1.3 table

Hubble中的表和其他数据库中的表一样，是主要存储数据的地方。按关系型数据库的形式组织数据，即按行和列来组织存储数据。

hive中的表
操作存储在hive中的表与hive兼容，包括支持分区分桶表等（临时表部分不兼容），操作语法略有差异，详情请参考第3章中的SQL语法详细内容。

hubble中的表
hubble中的表存在一些自定义的属性，其他部分与普通表无差异，操作语法详见第3章。

其他数据库中的表
原则上hubble对于其他数据库中的表做只读处理，这是为了防止篡改原有数据库，只做读取操作。对原有应用使用数据库起到保护作用。

对于所有的表，hubble在数据类型方面做了类型转换，使用SQL标准数据类型。

3.2 SQL基本语法

Hubble cli模式下执行

3.2.1 显示所有数据源

```
show catalogs;
```

3.2.2 创建数据库

```
create database hubble.test;
```

3.2.3 使用库

```
use hubble.test;
```

3.2.4 建表

```
#创建一张表
drop table if exists hubble.test.test;
create table hubble.test.test (
uid int,
createtime date,
endtime date,
cardid decimal,
name varchar(50),
age int,
note varchar(4)
)
with(shard_counts=1, index_define='{name="index_test",columns="uid",pk="false"}');

#创建一张表并建立多个索引
drop table if exists hubble.test.test;
create table hubble.test.test (
uid int,
createtime date,
endtime date,
cardid decimal,
name varchar(50),
age int,
note varchar(4)
)
with(shard_counts=1, index_define='[{name="index_test1",columns="uid",pk="false"},
{name="index_test2",columns="uid,name",pk="false"}]');

#创建表时，写入数据
create table hubble.test.test2 as select * from hubble.test.test;
```

3.2.5插入数据

```
#插入单条
insert into test(uid,name) values(111,'aaa');
#插入多条
insert into test(uid,name) values(222,'aaa'),(333,'bbb'),(444,'ccc');
```

3.2.6 删表操作

```
#删除表
drop table test;

#查看表是否删除成功
show tables;
```

3.3 执行脚本方式调用hubble

```
#hubble直接执行sql语句
./hubble-sql.sh -e "show catalogs"

#hubble执行sql脚本文件
vi test.sql
show catalogs;

./hubble-sql.sh -f ~/test.sql
```

3.4 查看hubble支持函数及使用说明

```
hubble cli执行 show functions;
```

3.5 说明

```
hubble支持标准SQL。参见Hubble数据库SQL语法文档
```

4.Hubble数据库的操作流程

4.1 Hubble数据库的OLTP操作流程

4.1.1 jdbc连接示例

```
    /**
     * hubble jdbc 执行sql操作jdbc示例
     * 需要加载hubble驱动包
     * @param args
     */
    public static void main(String[] args) {
        // 声明Connection对象
        Connection conn;
        // 驱动程序名
        String driver = "com.beagledata.hubble.jdbc.HubbleDriver";
        // URL指向要访问的数据库名mydata
        String url = "jdbc:hubble://192.168.118.101:30008/hubble";
        // Hubble配置时的用户名
        String user = "hadoop";
        // Hubble配置时的密码
```

```
        String password = "";

        String sql = "show catalogs";
        executeSql(driver, url, user, password, sql);
    }

    private static void executeSql(String driver, String url, String user, String
password, String sql) {
        Connection conn;
        try {
            // 加载驱动程序
            Class.forName(driver);
            conn = DriverManager.getConnection(url, user, password);
            if (!conn.isClosed())
                System.out.println("Succeeded connecting to the Database!");
            // 2.创建statement类对象，用来执行SQL语句！！
            Statement statement = conn.createStatement();

            // 3.执行sql
            boolean execute = statement.execute(sql);
            System.out.println("execute sql : " + execute);

            // 4.关闭连接
            statement.close();
            conn.close();
        } catch (ClassNotFoundException e) {
            System.out.println("Sorry,can`t find the Driver!");
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("数据库op成功！！");
        }
```

4.1.2 集成框架，应用开发

```
#集成spring+mybatis框架，提供外部服务
demo示例地址：
    https://github.com/longw5/hubble-mybatis

#框架集成，数据源配置，使用的druid连接池
    <bean id="jdbcDataSource" class="com.alibaba.druid.pool.DruidDataSource"
        destroy-method="close">
        <property name="driverClassName" value="${jdbc.driver}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.uid}" />
        <property name="password" value="${jdbc.pwd}" />
    </bean>

#db.properties配置
jdbc.driver=com.beagledata.hubble.jdbc.HubbleDriver
```

```
jdbc.url=jdbc:hubble://192.168.1.215:31009
jdbc.uid=hadoop
jdbc.pwd=

#mapper文件配置
    <insert id="insertsql" statementType="STATEMENT" parameterType="java.lang.String">
         ${_parameter}
    </insert>
```

4.2 Hubble数据库的OLAP操作流程

4.2.1 TPC-H使用生成大量数据数据

```
#参考官网
    http://www.tpc.org/tpch/ 用户使用手册
```

4.2.2 数据加载流程

4.2.2.1 支持实时查询，加载到hubble库

```
#全部操作脚本
#!/bin/bash

#配置hadoop环境
export HADOOP_HOME=/opt/hadoop/bin
#配置hubble安装目录
export HUBBLE_HOME=/opt/hubble_julong
#配置hive安装目录
export HIVE_HOME=/opt/hive
#配置数据目录
export DATA_DIR=/data/example/simple

#创建本地测试目录
mkdir -p ${DATA_DIR}/;

#生成测试数据
#echo -e "1,张三,24,男\n2,李四,26,男\n3,王五,56,男\n4,寒梅,18,女\n5,李蕾,15,女" >
${DATA_DIR}/simple_example.dat

#创建Hdfs上传目录
${HADOOP_HOME}/hdfs dfs -mkdir -p ${DATA_DIR};

#上传本地文件到hdfs
${HADOOP_HOME}/hdfs dfs -put ${DATA_DIR}/simple_example.dat ${DATA_DIR}/;

#查看文件是否上传成功
${HADOOP_HOME}/hdfs dfs -ls ${DATA_DIR};

#执行hive建库建表sql
${HIVE_HOME}/bin/hive -e "\
create database if not exists example;\
```

```
drop table if exists example.simple_example;\
create external table example.simple_example(
rowid int,name string,age int,gender string)\
row format delimited fields terminated by ',' STORED AS textfile;\
use example;\
show tables;\
desc simple_example";

#执行数据加载
${HIVE_HOME}/bin/hive -e "
LOAD DATA INPATH '${DATA_DIR}/' INTO TABLE example.simple_example";

#验证是否导入成功
${HIVE_HOME}/bin/hive -e "select * from example.simple_example";

#数据导入到hubble中
${HUBBLE_HOME}/bin/hubble-sql.sh -e "\
create database if not exists hubble.example_orc;\  #hubble中建库
drop table if exists hubble.example_orc.simple_example_orc;\    #删表
create table hubble.example_orc.simple_example_orc(\    #建表
rowid int,name varchar,age int,gender varchar) \
with(shard_counts=1, index_define='{name="index_rowid",columns="rowid",pk="false"}');\
use hubble.example_orc;\    #使用库
show tables;\   #show 表
desc simple_example_orc" #查看表结构

#执行数据转换操作
${HUBBLE_HOME}/bin/hubble-sql.sh -e "\
set session hubble.load_data_module=true;\
set session task_writer_count=16;\
insert into hive.example_orc.simple_example_orc \
select * from hive.example.simple_example";

#验证是否转换成功
${HUBBLE_HOME}/bin/hubble-sql.sh -e "select * from hive.example_orc.simple_example_orc";
```

4.2.2.2 支持统计分析查询，加载到hive库

```
#!/bin/bash

#配置hadoop环境
export HADOOP_HOME=/opt/hadoop/bin
#配置hubble安装目录
export HUBBLE_HOME=/opt/hubble_julong
#配置hive安装目录
export HIVE_HOME=/opt/hive
#配置数据目录
export DATA_DIR=/data/example/simple

#创建本地测试目录
mkdir -p ${DATA_DIR}/;

#生成测试数据
```

```
#echo -e "1,张三,24,男\n2,李四,26,男\n3,王五,56,男\n4,寒梅,18,女\n5,李蕾,15,女" >
${DATA_DIR}/simple_example.dat

#创建Hdfs上传目录
${HADOOP_HOME}/hdfs dfs -mkdir -p ${DATA_DIR};

#上传本地文件到hdfs
${HADOOP_HOME}/hdfs dfs -put ${DATA_DIR}/simple_example.dat ${DATA_DIR}/;

#查看文件是否上传成功
${HADOOP_HOME}/hdfs dfs -ls ${DATA_DIR};

#执行hive建库建表sql
${HIVE_HOME}/bin/hive -e "\
create database if not exists example;\
drop table if exists example.simple_example;\
create external table example.simple_example(\
rowid int,name string,age int,gender string)\
row format delimited fields terminated by ',' STORED AS textfile;\
use example;\
show tables;\
desc simple_example";

#执行数据加载
${HIVE_HOME}/bin/hive -e "
LOAD DATA INPATH '${DATA_DIR}/' INTO TABLE example.simple_example";

#验证是否导入成功
${HIVE_HOME}/bin/hive -e "select * from example.simple_example";

#执行hive orc格式建库建表sql
${HIVE_HOME}/bin/hive -e "\
create database if not exists example_orc;\     #hive中建库
drop table if exists example_orc.simple_example_orc;\   #删表
create table example_orc.simple_example_orc(\   @建表
rowid int,name string,age int,gender string)\
row format delimited fields terminated by '/001' STORED AS orc;\
use example_orc;\
show tables;\   #浏览表
desc simple_example_orc"    #查看表结构

#执行数据转换操作
${HUBBLE_HOME}/bin/hubble-sql.sh -e "\
set session hubble.load_data_module=true;\
set session task_writer_count=16;\
insert into hive.example_orc.simple_example_orc\
select * from hive.example.simple_example";

#验证是否转换成功
${HUBBLE_HOME}/bin/hubble-sql.sh -e "select * from hive.example_orc.simple_example_orc";
```

4.2.2.3 使用总结

```
AP的数据一般存放在hive目录下，转换成orc格式
TP的数据一般存放在hubble目录下。
```

5.hubble的运行维护

一些常见的集群运行问题

```
1.查看节点运行状态
    select * from system.runtime.nodes;

2.zookeeper与hubble节点通信session过期
    1.执行date命令，查看时间是否同步
    2.重启集群
```

6.Hubble数据库使用场景及案例

6.1 使用场景介绍

```
Hubble适用的常见场景
1.替换hive，做数据分析
2.大数据下的实时查询
3.实时联机交易
```

6.2 实时处理案例

6.3 离线实施案例

7.FAQ

8.练习

要求：

```
1.hubble数据的安装、部署、参数调整
2.使用hubble的各种管理页面、客户端工具
3.查看hubble运行状态、宕机重启
4.hubble的数据处理流程
```