第 1 章 Hive

1.1 Hive 安装部署

1）把 hive-3.1.3.tar.gz 上传到 linux 的/opt/software 目录下

2）解压 hive-3.1.3.tar.gz 到/opt/module/目录下面

```
[iflytek@hadoop102 software]$ tar -zxvf /opt/software/ apache-hive-3.1.3.tar.gz -C /opt/module/
```

3）修改 hive-3.1.3-bin.tar.gz 的名称为 hive

```
[iflytek@hadoop102 software]$ mv /opt/module/apache-hive-3.1.3-bin/ /opt/module/hive
```

4）修改/etc/profile.d/my_env.sh，添加环境变量

```
[iflytek@hadoop102 software]$ sudo vim /etc/profile.d/my_env.sh
```

添加内容

```
#HIVE_HOME
export HIVE_HOME=/opt/module/hive
export PATH=$PATH:$HIVE_HOME/bin
```

重启 Xshell 对话框或者 source 一下 /etc/profile.d/my_env.sh 文件，使环境变量生效。

```
[iflytek@hadoop102 software]$ source /etc/profile.d/my_env.sh
```

5）解决日志 Jar 包冲突，进入/opt/module/hive/lib 目录

```
[iflytek@hadoop102 lib]$ mv log4j-slf4j-impl-2.17.1.jar log4j-slf4j-impl-2.17.1.jar.bak
```

1.2 Hive 元数据配置到 MySQL

1.2.1 拷贝驱动

将 MySQL 的 JDBC 驱动拷贝到 Hive 的 lib 目录下。

```
[iflytek@hadoop102 lib]$ cp /opt/software/mysql/mysql-connector-j-8.0.31.jar /opt/module/hive/lib/
```

1.2.2 配置 Metastore 到 MySQL

在$HIVE_HOME/conf 目录下新建 hive-site.xml 文件。

```
[iflytek@hadoop102 conf]$ vim hive-site.xml
```

添加如下内容。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <!--配置 Hive 保存元数据信息所需的 MySQL URL 地址-->
    <property>
        <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?useSSL=false&amp;useUnicode=true&amp;characterEncoding=UTF-8&amp;allowPublicKeyRetrieval=true</value>
    </property>
```

```
    <!--配置 Hive 连接 MySQL 的驱动全类名-->
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.cj.jdbc.Driver</value>
    </property>

    <!--配置 Hive 连接 MySQL 的用户名 -->
    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>

    <!--配置 Hive 连接 MySQL 的密码 -->
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>000000</value>
    </property>

    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
    </property>

    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>

    <property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
    </property>

    <property>
        <name>hive.server2.thrift.bind.host</name>
        <value>hadoop102</value>
    </property>

    <property>
        <name>hive.metastore.event.db.notification.api.auth</name>
        <value>false</value>
    </property>

    <property>
        <name>hive.cli.print.header</name>
        <value>true</value>
    </property>

    <property>
        <name>hive.cli.print.current.db</name>
        <value>true</value>
    </property>
</configuration>
```

1.3 启动 Hive

1.3.1 初始化元数据库

1）登陆 MySQL

```
[iflytek@hadoop102 conf]$ mysql -uroot -p000000
```

2）新建 Hive 元数据库

```
mysql> create database metastore;
```

3）初始化 Hive 元数据库

```
[iflytek@hadoop102 hive]$ bin/schematool -initSchema -dbType mysql
-verbose
```

4）修改元数据库字符集

Hive 元数据库的字符集默认为 Latin1，由于其不支持中文字符，所以建表语句中如果包含中文注释，会出现乱码现象。如需解决乱码问题，须做以下修改。

修改 Hive 元数据库中存储注释的字段的字符集为 utf-8。

（1）字段注释

```
mysql> use metastore;
mysql> alter table COLUMNS_V2 modify column COMMENT varchar(256)
character set utf8;
```

（2）表注释

```
mysql> alter table TABLE_PARAMS modify column PARAM_VALUE
mediumtext character set utf8;
```

4）退出 mysql

```
mysql> quit;
```

1.3.2 启动 Hive 客户端

注意：在启动 hive 之前群起 HDFS

1）启动 Hive 客户端

```
[iflytek@hadoop102 hive]$ hive
```

2）查看一下数据库

```
hive (default)> show databases;
OK
database_name
default
Time taken: 0.955 seconds, Fetched: 1 row(s)
```

1.3.3 hiveserver2 服务

1）hiveserver2 部署

（1）Hadoop 端配置

hivesever2 的模拟用户功能，依赖于 Hadoop 提供的 proxy user（代理用户功能），只有 Hadoop

中的代理用户才能模拟其他用户的身份访问 Hadoop 集群。因此，需要将 hiveserver2 的启动用户设置为 Hadoop 的代理用户，配置方式如下：

修改配置文件 core-site.xml，然后记得分发三台机器

```
[iflytek@hadoop102 ~]$ cd $HADOOP_HOME/etc/hadoop
[iflytek@hadoop102 hadoop]$ vim core-site.xml
```

增加如下配置：

```
<!--配置所有节点的 iflytek 用户都可作为代理用户-->
<property>
    <name>hadoop.proxyuser.iflytek.hosts</name>
    <value>*</value>
</property>

<!--配置 iflytek 用户能够代理的用户组为任意组-->
<property>
    <name>hadoop.proxyuser.iflytek.groups</name>
    <value>*</value>
</property>

<!--配置 iflytek 用户能够代理的用户为任意用户-->
<property>
    <name>hadoop.proxyuser.iflytek.users</name>
    <value>*</value>
</property>
```

（2）Hive 端配置

在 hive-site.xml 文件中添加如下配置信息

```
[iflytek@hadoop102 conf]$ vim hive-site.xml

<!-- 指定 hiveserver2 连接的 host -->
<property>
    <name>hive.server2.thrift.bind.host</name>
    <value>hadoop102</value>
</property>

<!-- 指定 hiveserver2 连接的端口号 -->
<property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
</property>
```

3）测试

（1）启动 hiveserver2

```
[iflytek@hadoop102 hive]$ bin/hive --service hiveserver2
```

（2）使用命令行客户端 beeline 进行远程访问

启动 beeline 客户端

```
[iflytek@hadoop102 hive]$ bin/beeline -u
jdbc:hive2://hadoop102:10000 -n iflytek
```

看到如下界面

```
Connecting to jdbc:hive2://hadoop102:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://hadoop102:10000>
```

1.3.4 编写 Hive 服务启动脚本（了解）

2）为了方便使用，可以直接编写脚本来管理服务的启动和关闭

```
[iflytek@hadoop102 hive]$ cd
[iflytek@hadoop102 hive]$ vim bin/hiveservices.sh
```

内容如下：此脚本的编写不要求掌握。直接拿来使用即可。

```
#!/bin/bash

HIVE_LOG_DIR=$HIVE_HOME/logs
if [ ! -d $HIVE_LOG_DIR ]
then
    mkdir -p $HIVE_LOG_DIR
fi

#检查进程是否运行正常，参数 1 为进程名，参数 2 为进程端口
function check_process()
{
    pid=$(ps -ef 2>/dev/null | grep -v grep | grep -i $1 | awk
'{print $2}')
    ppid=$(netstat -nltp 2>/dev/null | grep $2 | awk '{print $7}' |
cut -d '/' -f 1)
    echo $pid
    [[ "$pid" =~ "$ppid" ]] && [ "$ppid" ] && return 0 || return 1
}

function hive_start()
{
    server2pid=$(check_process HiveServer2 10000)
    cmd="nohup                    hive                    --service
hiveserver2 >$HIVE_LOG_DIR/hiveServer2.log 2>&1 &"
    [ -z "$server2pid" ] && eval $cmd || echo "HiveServer2 服务已启动
"
}

function hive_stop()
{
    server2pid=$(check_process HiveServer2 10000)
    [ "$server2pid" ] && kill $server2pid || echo "HiveServer2 服务
未启动"
}

case $1 in
"start")
    hive_start
    ;;
"stop")
    hive_stop
    ;;
"restart")
    hive_stop
    sleep 2
    hive_start
    ;;
"status")
    check_process HiveServer2 10000 >/dev/null && echo "HiveServer2
服务运行正常" || echo "HiveServer2 服务运行异常"
```

```
    ;;
*)
    echo Invalid Args!
    echo 'Usage: '$(basename $0)' start|stop|restart|status'
    ;;
esac
```

3）添加执行权限

```
[iflytek@hadoop102 hive]$ chmod +x hiveservices.sh
```

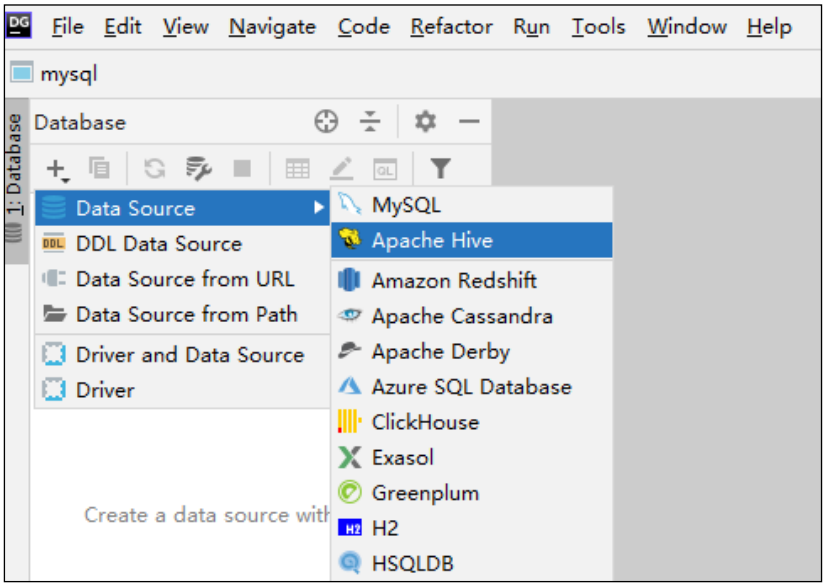
4）启动 Hive 后台服务

```
[iflytek@hadoop102 hive]$ hiveservices.sh start
```

第 2 章 配置 DataGrip 连接

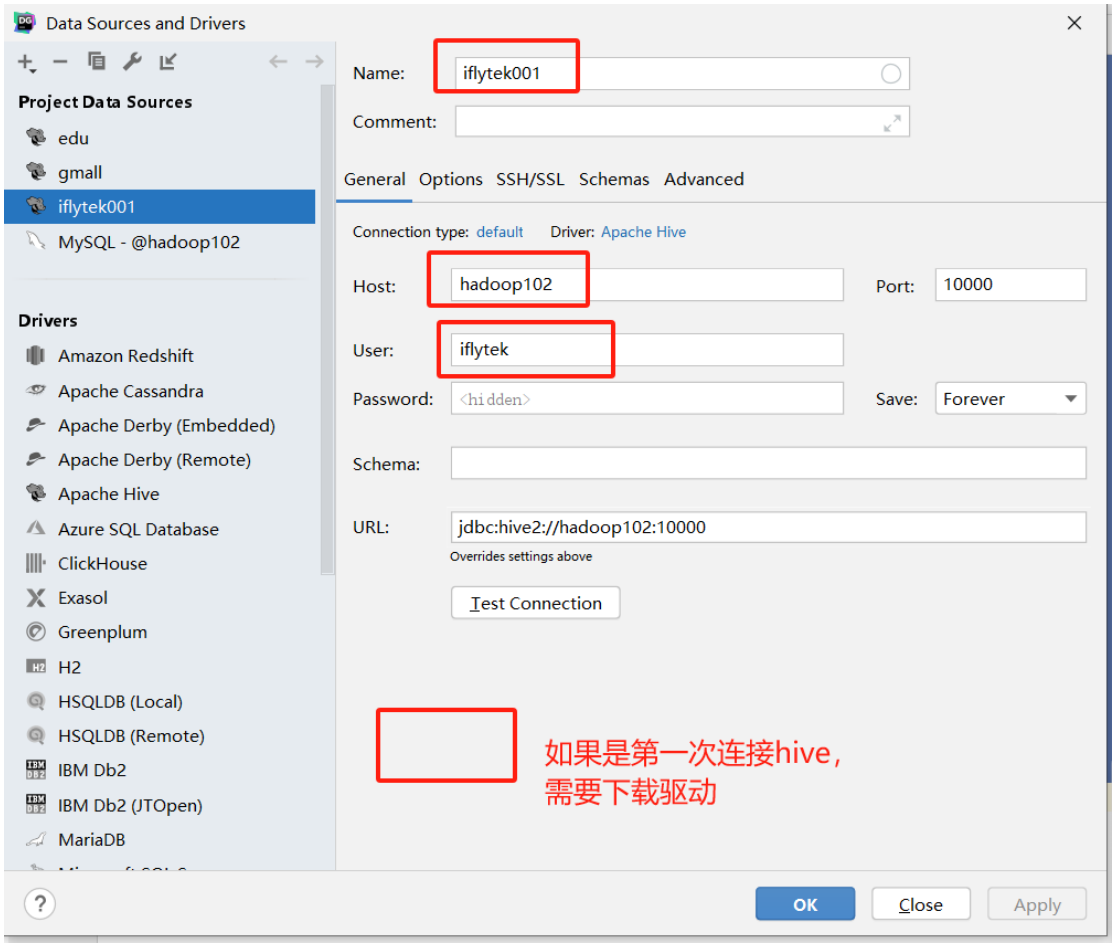
安装步骤傻瓜式安装

（1）创建连接

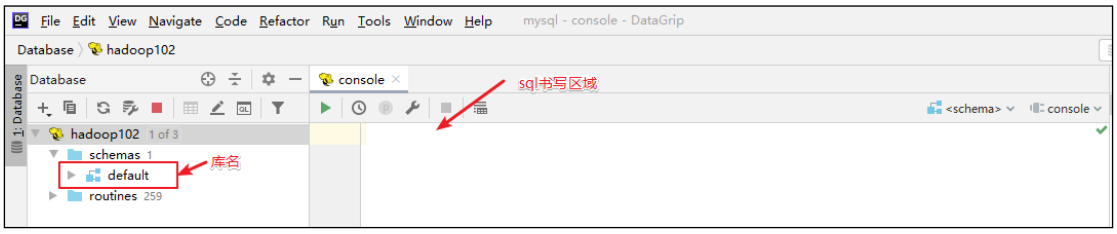


（2）配置连接属性

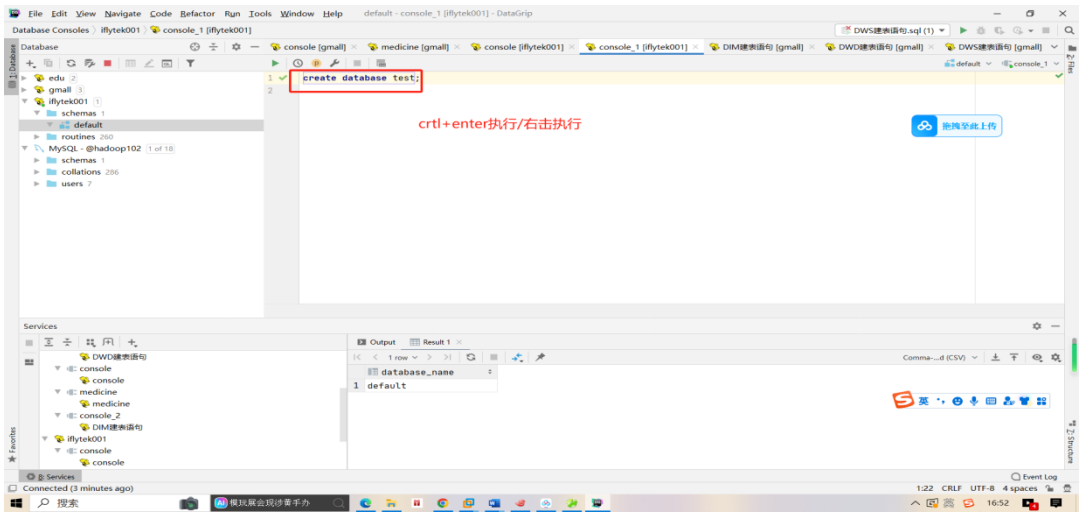
所有属性配置，和 Hive 的 beeline 客户端配置一致即可。初次使用，配置过程会提示缺少 JDBC 驱动，按照提示下载即可。



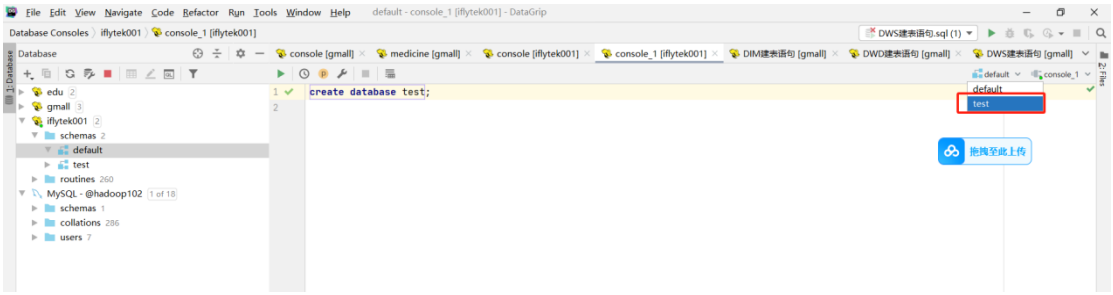
（3）界面介绍



（4）测试 sql 执行



（5）修改数据库



第 3 章 Hive 建表和 Mysql 生成数据上传 HDFS

3.1 Hive 建表

在上传 Mysql 数据之前需要在 Hive 中先建立对应的表

1） 药材表建表语句

```
DROP TABLE IF EXISTS medicine;
create external table if not exists medicine(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    approval_code string COMMENT '药物批号',
    dose_type string COMMENT '剂量',
    name string COMMENT '药物名称',
    name_en string COMMENT '英文名称',
    price string COMMENT '药品价格',
    specs string COMMENT '规格',
    trade_name string COMMENT '商品表'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/medicine';
```

2） 就诊表建表语句

```
DROP TABLE IF EXISTS consultation;
create external table if not exists consultation(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    consultation_fee string COMMENT '咨询费用',
    description string COMMENT '病情描述',
    diagnosis string COMMENT '诊断',
    rating string COMMENT '评分：1-5 分',
    review string COMMENT '评价',
    status string COMMENT '状态：201.未填写病情描述 202.未支付 203.已支付 204.正在诊断 205.诊断结束 206.已出处方 207.已评价',
    doctor_id string COMMENT '接诊医生',
    patient_id string COMMENT '就诊人',
    user_id string COMMENT '用户'
)
```

```
row format delimited fields terminated by ','
stored as textfile
location '/medical/consultation';
```

3）字典表建表语句

```
DROP TABLE IF EXISTS dict;
create external table if not exists dict(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    value string COMMENT '值'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/dict';
```

4）医生表建表语句

```
DROP TABLE IF EXISTS doctor;
create external table if not exists doctor(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    birthday string COMMENT '出生日期',
    consultation_fee string COMMENT '就诊费用',
    gender string COMMENT '性别：101.男 102.女',
    name string COMMENT '姓名',
    specialty string COMMENT '专业：详情见字典表 5xx 条目',
    title string COMMENT '职称：301. 医士 302. 医师 303. 主治医师 304.
副主任医师 305. 主任医师',
    hospital_id string COMMENT '所属医院'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/doctor';
```

5）医院建表语句

```
DROP TABLE IF EXISTS hospital;
create external table if not exists hospital(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    address string COMMENT '地址',
    alias string COMMENT '医院别名',
    bed_num string COMMENT '病床数量',
    city string COMMENT '市',
    department_num string COMMENT '科室数量',
    district string COMMENT '区县',
    establish_time string COMMENT '建立时间',
    health_care_num string COMMENT '医护人数',
    insurance string COMMENT '是否医保',
    level string COMMENT '医院级别，一级甲等，二级甲等',
    name string COMMENT '医院名称',
    province string COMMENT '省（直辖市）'
)
row format delimited fields terminated by ','
stored as textfile
```

```
location '/medical/hospital';
```

6）就诊人建表语句

```
DROP TABLE IF EXISTS patient;
create external table if not exists patient(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    birthday string COMMENT '出生日期',
    gender string COMMENT '性别',
    name string COMMENT '姓名',
    user_id string COMMENT '所属用户'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/patient';
```

7）支付表建表语句

```
DROP TABLE IF EXISTS payment;
create external table if not exists payment(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    payment_amount string COMMENT '支付金额',
    status string COMMENT '支付状态：202.未支付 203.已支付',
    consultation_id string COMMENT '关联就诊记录',
    prescription_id string COMMENT '关联处方',
    user_id string COMMENT '支付用户'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/payment';
```

8）处方表建表语句

```
DROP TABLE IF EXISTS prescription;
create external table if not exists prescription(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '患者修改时间',
    instruction string COMMENT '处方说明',
    status string COMMENT '处方状态：202.未支付 203.已支付',
    total_amount string COMMENT '总金额',
    consultation_id string COMMENT '就诊记录',
    doctor_id string COMMENT '医生',
    patient_id string COMMENT '患者'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/prescription';
```

9）处方详情建表语句

```
DROP TABLE IF EXISTS prescription_detail;
create external table if not exists prescription_detail(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    count string COMMENT '剂量',
```

```
    instruction string COMMENT '服用说明',
    medicine_id string COMMENT '药品',
    prescription_id string COMMENT '所属处方'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/prescription_detail';
```

10）　用户建表语句

注：如果报错，尝试 user 替换为`medical.user`

/medical/user 不改变

```
DROP TABLE IF EXISTS user;
create external table if not exists user(
    id string COMMENT '主键',
    create_time string COMMENT '创建时间',
    update_time string COMMENT '修改时间',
    email string COMMENT '电邮',
    hashed_password string COMMENT '密码',
    telephone string COMMENT '电话',
    username string COMMENT '用户名'
)
row format delimited fields terminated by ','
stored as textfile
location '/medical/user';
```

3.2 Mysql 生成数据上传 HDFS

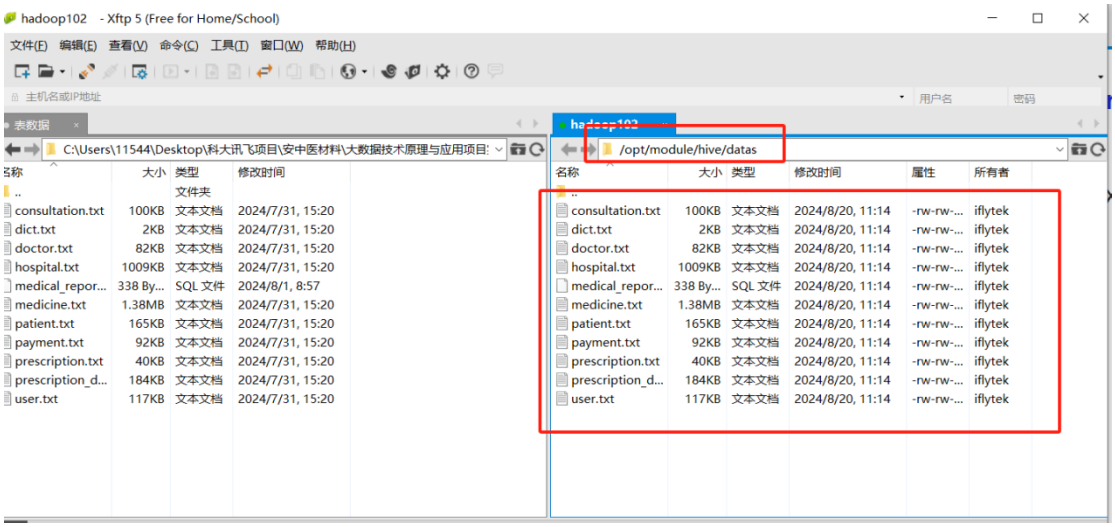
1）准备好导出的数据

consultation.txt	2024/7/31 15:20	文本文档	101 KB
dict.txt	2024/7/31 15:20	文本文档	3 KB
doctor.txt	2024/7/31 15:20	文本文档	83 KB
hospital.txt	2024/7/31 15:20	文本文档	1,010 KB
medical_report.sql	2024/8/1 8:57	SQL 文件	1 KB
medicine.txt	2024/7/31 15:20	文本文档	1,411 KB
patient.txt	2024/7/31 15:20	文本文档	166 KB
payment.txt	2024/7/31 15:20	文本文档	93 KB
prescription.txt	2024/7/31 15:20	文本文档	41 KB
prescription_detail.txt	2024/7/31 15:20	文本文档	185 KB
user.txt	2024/7/31 15:20	文本文档	118 KB

2）创建目录

```
[iflytek@hadoop102 conf]$ mkdir /opt/module/hive/datas
```

3）上传数据到该目录



4）启动 HDFS 集群

6）逐个文件上传，插入对应的 Hive 表中，命令如下

- 药材表上传命令

```
load data local inpath '/opt/module/hive/datas/medicine.txt' into table medicine;
```

- 就诊表上传命令

```
load data local inpath '/opt/module/hive/datas/consultation.txt' into table consultation;
```

- 字典表上传命令

```
load data local inpath '/opt/module/hive/datas/dict.txt' into table dict;
```

- 医生表上传命令

```
load data local inpath '/opt/module/hive/datas/doctor.txt' into table doctor;
```

- 医院表上传命令

```
load data local inpath '/opt/module/hive/datas/hospital.txt' into table hospital;
```

- 就诊人表上传命令

```
load data local inpath '/opt/module/hive/datas/patient.txt' into table patient;
```

- 支付表上传命令

```
load data local inpath '/opt/module/hive/datas/payment.txt' into table payment;
```

- 处方表上传命令

```
load data local inpath '/opt/module/hive/datas/prescription.txt' into table prescription;
```

- 处方详情表上传命令

```
load data local inpath '/opt/module/hive/datas/prescription_detail.txt' into table prescription_detail;
```

- 用户表上传命令

```
load data local inpath '/opt/module/hive/datas/user.txt' into table user;
```

第 4 章 Hive 数据分析

需求 1：统计不同省份中医院数量

建表语句

```
DROP TABLE IF EXISTS province_medicine_hospital_num;
CREATE EXTERNAL TABLE province_medicine_hospital_num
(
    `province`        STRING COMMENT '省份名称',
    `med_hos_num` STRING COMMENT '中医药数量'
```

```
) COMMENT '不同省份中医药数量'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/province_medicine_hospital_num';
```

分析语句

```
insert overwrite table province_medicine_hospital_num
select
     province,
     count(1) as med_hos_num
from hospital
where name like '%中医%'
group by province
```

需求 2 统计所有中医院不同职称医生的数量

建表语句

```
DROP TABLE IF EXISTS all_hospital_doctor_titile_num;
CREATE EXTERNAL TABLE all_hospital_doctor_titile_num
(
    `title`        STRING COMMENT '职称',
    `doctor_num` STRING COMMENT '医生数量'
) COMMENT '所有中医院不同职称医生数量'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/all_hospital_doctor_titile_num';
```

分析语句

```
insert overwrite table all_hospital_doctor_titile_num
select
      dict.value as title,
      count(1) as doctor_num
from
(
select
    id,
    name
from hospital
where name like '%中医%'
    ) as hospital
join (
       select
          title,
       hospital_id
       from doctor
       ) as doctor
on hospital.id = doctor.hospital_id
join (
    select
        id,
        value
    from dict
    ) as dict
on doctor.title=dict.id
group by dict.value
```

需求 3 统计每个类型的药品数量

建表语句

```
DROP TABLE IF EXISTS type_medicine_num;
CREATE EXTERNAL TABLE type_medicine_num
(
    `dose_type`        STRING COMMENT '药物分类',
    `medicine_num` STRING COMMENT '药品种类数'
) COMMENT '统计每个类型的药品数量'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/type_medicine_num';
```

分析语句

```
insert overwrite table type_medicine_num
select
     dose_type,
      count(1) as medicine_num
from medicine
group by dose_type
```

需求四 统计历史以来不同中医院的病人数

建表语句

```
DROP TABLE IF EXISTS history_medical_hospital_patient_num;
CREATE EXTERNAL TABLE history_medical_hospital_patient_num
(
    `name`        STRING COMMENT '中医院名称',
    `patient_num` STRING COMMENT '病人数'
) COMMENT '统计历史以来不同中医院的病人数'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/history_medical_hospital_patient_num';
```

分析语句

```
insert overwrite table history_medical_hospital_patient_num
select hospital.name,
      count(1) as patient_num
from (
        select doctor_id,
               patient_id
        from consultation
        group by doctor_id, patient_id
    ) as doc_pat
join (
    select id,
          hospital_id
    from doctor
) as doctor
on doc_pat.doctor_id=doctor.id
join(
    select id,
          name
    from hospital
    where name like '%中医%'
) as hospital
on doctor.hospital_id=hospital.id
group by hospital.name
```

需求 5 统计历史以来不同中医院就诊次数

建表语句

```
DROP TABLE IF EXISTS history_medical_hospital_consultation_num;
CREATE EXTERNAL TABLE history_medical_hospital_consultation_num
(
    `name`        STRING COMMENT '中医院名称',
    `consultation_num` STRING COMMENT '就诊次数'
) COMMENT '统计历史以来不同中医院就诊次数'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/history_medical_hospital_consultation_num';
```

分析语句

```
insert overwrite table history_medical_hospital_consultation_num
select hospital.name,
      count(1) as consultation_num
from (
        select doctor_id
        from consultation
    ) as doc
join (
    select id,
          hospital_id
    from doctor
) as doctor
on doc.doctor_id=doctor.id
join(
    select id,
          name
    from hospital
    where name like '%中医%'
) as hospital
on doctor.hospital_id=hospital.id
group by hospital.name
```

需求 6 统计历史以来中医院收入 TOP 5

建表语句

```
DROP TABLE IF EXISTS history_medical_hospital_amount_top5;
CREATE EXTERNAL TABLE history_medical_hospital_amount_top5
(
    `name`        STRING COMMENT '中医院名称',
    `total_amount` STRING COMMENT '总金额'
) COMMENT '统计历史以来中医院收入 TOP 5'
row format delimited fields terminated by '\t'
stored as textfile
location '/medical/history_medical_hospital_amount_top5';
```

分析语句

```
insert overwrite table history_medical_hospital_amount_top5
select
    hospital.name,
    round(sum(total_amount),1) as total_amount
from (
        select doctor_id,
              total_amount
        from prescription
        where status = '203'
```

```
    ) as doc
join (
    select id,
           hospital_id
    from doctor
) as doctor
on doc.doctor_id=doctor.id
join(
    select id,
           name
    from hospital
    where name like '%中医%'
) as hospital
on doctor.hospital_id=hospital.id
group by hospital.name
sort by total_amount desc
limit 5
```