一 内容回顾（列举前一天重点难点内容）

1.1 教学重点

```
Flume配置
Flume Agant结构
Flume Source
Flume Channel
Flume Sink
拦截器
```

二 教学目标

```
Sqoop的简介
Sqoop的底层原理
Sqoop的环境配置
Sqoop的配置安装
Sqoop导入导出
```

三 教学导读

3.1 产生背景

```
基于传统关系型数据库的稳定性，还是有很多企业将数据存储在关系型数据库中；早期由于工具的缺乏，Hadoop与传统数据库之间的数据传输非常困难。基于前两个方面的考虑，需要一个在传统关系型数据库和Hadoop之间进行数据传输的项目，Sqoop应运而生。
```

3.2 Sqoop是什么

Sqoop是一个用于**Hadoop**和**结构化数据存储**（如关系型数据库）之间进行高效传输大批量数据的工具。它包括以下两个方面： 可以使用Sqoop将数据从**关系型数据库管理系统(如MySql)导入到Hadoop系统**(如HDFS、Hive、HBase)中将数据从Hadoop系统中抽取并导出到关系型数据库(如MySql)

常见数据库开源工具：

1. Sqoop
2. Datax
3. Kettle
4. Cannal

四 教学内容

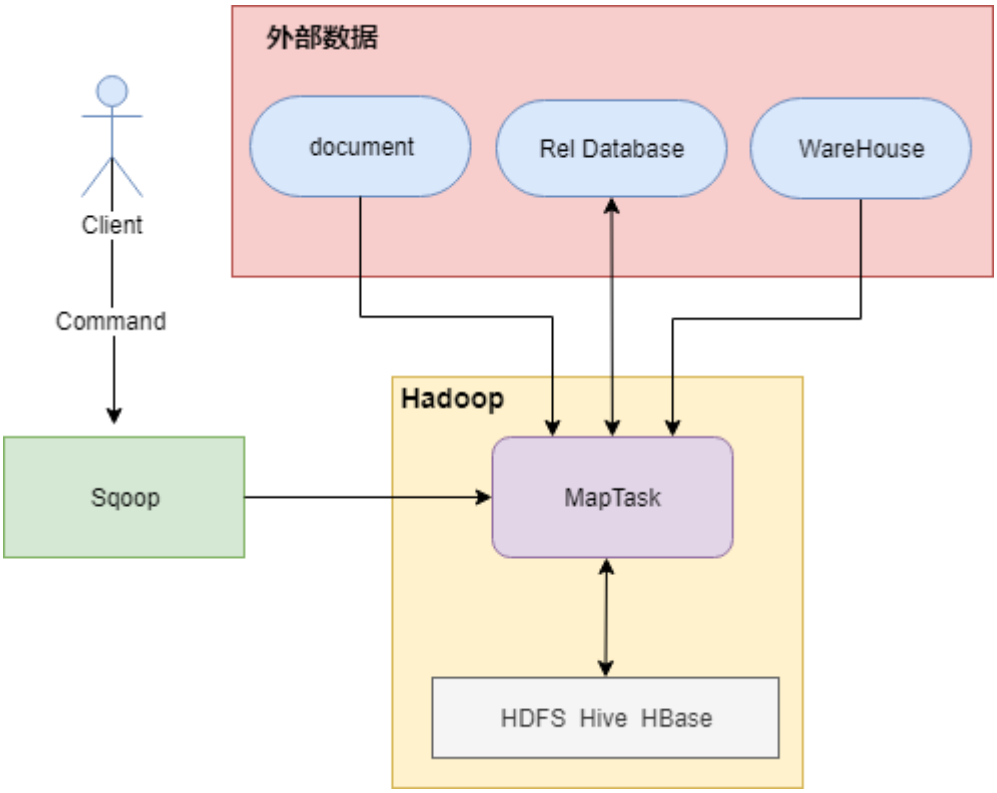
4.1 Sqoop简介以及使用

4.1.1 底层实现原理

Sqoop的**核心设计思想是利用MapReduce加快数据传输速度。**也就是说Sqoop的导入和导出功能是通过基于Map Task（只有map）的MapReduce作业实现的。所以它是一种批处理方式进行数据传输，难以实现实时的数据进行导入和导出。

> 官网介绍： Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.

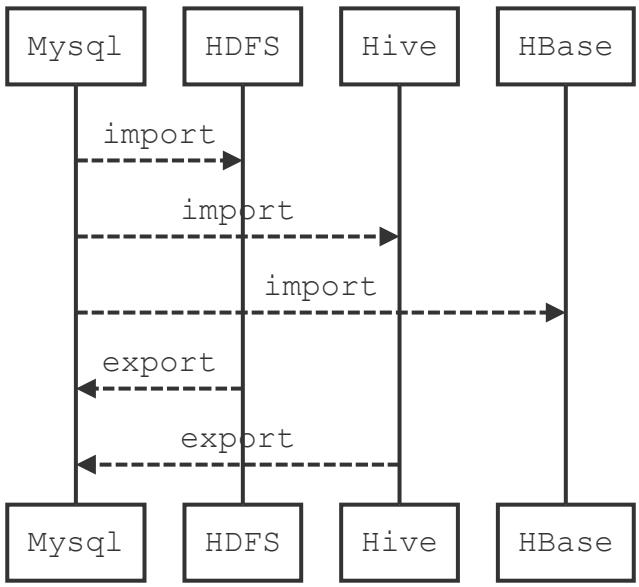
Sqoop结构图:



4.1.2 特点

- 优点：它可以将跨平台的数据进行整合。
- 缺点：它不是很灵活。

主要执行操作



Sqoop的重要的几个关键词

- import： 从关系型数据库到Hadoop
- export： 从Hadoop到关系型数据库。

4.2 Sqoop的安装

> 注意：在安装Sqoop之前要配置好本机的Java环境和Hadoop环境
>
> 先把Spoop的安装包 sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz 拷贝在系统目录下的 /root/softwares下面

4.2.1 解压配置环境变量

```
# 解压tar.gz包
[root@qianfeng01 local] tar -zxvf /root/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz -C /usr/local/

#把Sqoop的安装路径修改为sqoop-1.4.7,方便以后配置和调用
[root@qianfeng01 local]# mv  sqoop-1.4.7.bin__hadoop-2.6.0 sqoop-1.4.7
[root@qianfeng01 sqoop-1.4.7]# vi /etc/profile
# 追加内容如下:

export SQOOP_HOME=/usr/local/sqoop-1.4.7
export PATH=$PATH:$SQOOP_HOME/bin
```

4.2.2 新建配置文件

```
[root@qianfeng01 sqoop-1.4.7] mv ./conf/sqoop-env-template.sh ./conf/sqoop-env.sh
```

4.2.3 修改配置文件

配置文件:

```
[root@qianfeng01 sqoop-1.4.7]  vi ./conf/sqoop-env.sh
```

按照本系统实际安装的Hadoop系列目录配置好下面的路径:

```
export HADOOP_COMMON_HOME=/usr/local/hadoop-3.2.1
export HADOOP_MAPRED_HOME=/usr/local/hadoop-3.2.1
export HIVE_HOME=/usr/local/hive-3.1.2
export ZOOCFGDIR=/usr/local/zookeeper-3.6.2
```

4.2.4 拷贝MySql驱动

因为我们现在通过JDBC让MySql和HDFS等进行数据的导入导出,所以我们先必须把JDBC的驱动包拷贝到`sqoop/lib`路径下,如下

```
[root@qianfeng01 sqoop-1.4.7] cp /root/mysql-connector-java-5.1.18.jar ./lib/
```

4.2.5 验证安装

```
#查看Sqoop的版本
[root@qianfeng01 sqoop-1.4.7] sqoop version
```

4.3 Sqoop命令执行

4.3.1 常见命令执行参数

通过Sqoop加不同参数可以执行导入导出,通过`sqoop help`可以查看常见的命令行

```
#常见Sqoop参数
[root@qianfeng01 sqoop-1.4.7] sqoop help
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table #导出
  help               List available commands
  import             Import a table from a database to HDFS  #导入
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import mainframe datasets to HDFS
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  version            Display version information
```

4.3.2 直接执行命令

Sqoop运行的时候不需要启动后台进程,直接执行`sqoop`命令加参数即可.简单举例如下:

```
# #通过参数用下面查看数据库
[root@qianfeng01 sqoop-1.4.7] sqoop list-databases --connect jdbc:mysql://localhost:3306 --username root --password 123456;
```

4.3.3 通过文件传递参数(脚本)

在执行Sqoop命令时,如果每次执行的命令都相似,那么把相同的参数可以抽取出来,放在一个文本文件中,把执行时的参数加入到这个文本文件为参数即可. 这个文本文件可以用`--options-file`来指定,平时可以用定时任务来执行这个脚本,避免每次手工操作.

把**3.2章节**中命令中的JDBC连接的参数一般是不变的,可以把它抽取出来放在一个文件中`/.../sqoop-1.4.7/config.conf`,如下:

```
list-databases
--connect
jdbc:mysql://localhost:3306
--username
root
--password
123456
```

那么上面的执行的命令就可以变为:

```
[root@qianfeng01 sqoop-1.4.7] bin/sqoop --options-file config.conf
```

为了让配置文件config.conf的可读性更强,可以加入空行和注释,不会影响文件内容的读取,如下:

```
# 指令: 列出mysql中的所有数据库
list-databases

# 指定连接字符串
--connect
jdbc:mysql://localhost:3306

--username
root

--password
123456

```

4.3.4 Import 详解

import是从关系数据库导入到Hadoop,下面是一些通用参数介绍:

4.3.4.1 通用参数

如下:

Argument	Description
--connect	指定JDBC连接字符串
`--connection-manager`	指定连接管理类
`--driver`	指定连接的驱动程序
`-P`	从控制台读入密码(可以防止密码显示中控制台)
--password	指定访问数据库的密码
--username	指定访问数据库的用户名

4.3.4.1.1 连接数据库

Sqoop的设计就是把数据库数据导入HDFS,所以必须指定连接字符串才能访问数据库,这个连接字符串类似于URL,这个连接字符串通过`--connect`参数指定,它描述了连接的数据库地址和具体的连接数据库,譬如:

```
[root@qianfeng01 sqoop-1.4.7] sqoop import --connect
jdbc:mysql://database.example.com/employees
#指定连接的服务器地址是database.example.com ,要连接的数据库是employees
```

上面连接命令只是指定数据库,默认情况下数据库都是需要用户名和参数的,在这里可以用`--username`和`--password`来指定,譬如:

```
#指定用户名和密码来连接数据库
[root@qianfeng01 sqoop-1.4.7] sqoop import --connect jdbc:mysql://localhost:3306/mysql --
username root --password 123456;
```

4.3.4.1.2 查看数据库

在Sqoop中,可以通过`list-databases`参数来查看MySql的数据库,这样在导入之前可以得到所有的数据库的名字,具体案例如下:

```
# 列出所有数据库
[root@qianfeng01 sqoop-1.4.7] sqoop list-databases --connect jdbc:mysql://localhost:3306 --
username root --password 123456;
```

4.3.4.1.3 查看所有表

在得到所有数据库的名字后,也可以查看当前数据库中的所有表,可以使用`list-tables`参数来进行查看,查看的时候在url连接中一定要指定数据库的名字.

```
# 列出数据库中所有表
[root@qianfeng01 sqoop-1.4.7] sqoop list-tables --connect jdbc:mysql://localhost:3306/qfdb --
username root --password 123456;
```

4.3.4.2 Import的控制参数

常见import的控制参数有如下几个:

Argument	Description
`--append`	通过追加的方式导入到HDFS
`--as-avrodatafile`	导入为 Avro Data 文件格式
`--as-sequencefile`	导入为 SequenceFiles文件格式
`--as-textfile`	导入为文本格式 (默认值)
`--as-parquetfile`	导入为 Parquet 文件格式
`--columns`	指定要导入的列
`--delete-target-dir`	如果目标文件夹存在,则删除
`--fetch-size`	一次从数据库读取的数量大小
`-m,--num-mappers`	m 用来指定map tasks的数量,用来做并行导入
`-e,--query`	指定要查询的SQL语句
`--split-by`	用来指定分片的列
`--table`	需要导入的表名
`--target-dir`	HDFS 的目标文件夹
`--where`	用来指定导入数据的where条件
`-z,--compress`	是否要压缩
`--compression-codec`	使用Hadoop压缩 (默认是 gzip)

4.3.4.2.1 指定表导入

数据准备

在本地MySql数据库中新建一个 `qfdb` 数据库,sql代码在 `data/qfdb.sql` 中,如下:

```
CREATE TABLE emp(
    empno        INT primary key,
    ename        VARCHAR(50),
    job        VARCHAR(50),
    mgr        INT,
    hiredate    DATE,
    sal        DECIMAL(7,2),
    comm        decimal(7,2),
    deptno        INT
) ;
INSERT INTO emp values(7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);
```

```
INSERT INTO emp values(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);
INSERT INTO emp values(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);
INSERT INTO emp values(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);
INSERT INTO emp values(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);
INSERT INTO emp values(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);
INSERT INTO emp values(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);
INSERT INTO emp values(7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20);
INSERT INTO emp values(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);
INSERT INTO emp values(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);
INSERT INTO emp values(7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20);
INSERT INTO emp values(7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);
INSERT INTO emp values(7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);
INSERT INTO emp values(7934,'MILLER','CLERK',7782,'1982-01-23',1300,NULL,10);
```

Sqoop的典型导入都是把关系数据库中的表导入到HDFS中,使用`--table`参数可以指定具体的表导入到HDFS,譬如用`--table emp`,默认情况下是全部字段导入.如下:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp \
--delete-target-dir
```

可以快速使用hdfs的命令查询结果

```
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /sqoopdata/emp/par*
```

4.3.4.2.2 指定列导入

如果想导入某几列,可以使用`--columns`,如下:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp \
--columns 'empno,mgr' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp \
--delete-target-dir
```

可以使用下面hdfs命令快速查看结果

```
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /sqoopdata/emp/par*
```

4.3.4.2.3 指定条件导入

在导入表的时候,也可以通过指定where条件来导入,具体参数使用`--where`,譬如要导入员工号大于7800的记录,可以用下面参数:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp \
--columns 'empno,mgr' \
--where 'empno>7800' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/5 \
--delete-target-dir
```

用命令查询结果:

```
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /sqoopdata/emp/par*
```

结果如下:

```
7839,null
7844,7698
7876,7788
7900,7698
7902,7566
7934,7782
```

4.3.4.2.4 指定Sql导入

上面的可以通过表,字段,条件进行导入,但是还不够灵活,其实Sqoop还可以通过自定义的sql来进行导入,可以通过`--query`参数来进行导入,这样就最大化的用到了Sql的灵活性.如下:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--query 'select empno,mgr,job from emp WHERE empno>7800 and $CONDITIONS' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp \
--delete-target-dir \
--split-by empno \
-m 1
```

注意:在通过`--query`来导入数据时,必须要指定`--target-dir`

如果你想通过并行的方式导入结果，每个map task需要执行sql查询语句的副本，结果会根据Sqoop推测的边界条件分区。query必须包含`$CONDITIONS`。这样每个Sqoop程序都会被替换为一个独立的条件。同时你必须指定`--split-by`.分区

-m 1 是指定通过一个Mapper来执行流程

查询执行结果

```
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /sqoopdata/emp/par*
```

结果如下:

```
7839,null,PRESIDENT
7844,7698,SALESMAN
7876,7788,CLERK
7900,7698,CLERK
7902,7566,ANALYST
7934,7782,CLERK
```

4.3.4.2.5 单双引号区别

在导入数据时,默认的字符引号是单引号,这样Sqoop在解析的时候就安装字面量来解析,不会做转移:例如:

```
--query 'select empno,mgr,job from emp WHERE empno>7800 and $CONDITIONS'
```

如果使用了双引号,那么Sqoop在解析的时候会做转义的解析,这时候就必须要加转义字符\: 如下:

```
--query "select empno,mgr,job from emp WHERE empno>7800 and \$CONDITIONS"
```

4.3.4.2.6 MySql缺主键问题

1如果MySql的表没有主键，将会报错:

```
19/12/02 10:39:50 ERROR tool.ImportTool: Import
failed: No primary key could be found for table u1. Please specify one with
-- split-  by or perform a sequential import with '-m 1
```

解决方案:

```
通过 --split-by 来指定要分片的列
```

代码如下:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--query 'select empno,mgr,job from emp WHERE empno>7800 and $CONDITIONS' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp \
--delete-target-dir \
--split-by empno \
-m 1
```

4.3.4.3 导入到Hive中

4.3.4.3.1 说明

Sqoop的导入工具的主要功能是将数据上传到HDFS中的文件中。如果您有一个与`HDFS`集群相关联的Hive，`Sqoop`还可以通过生成和执行`CREATETABLE`语句来定义Hive中的数据，从而将数据导入到Hive中。将数据导入到Hive中就像在Sqoop命令行中添加--hive-import选项。

如果Hive表已经存在，则可以指定`--hive-overwrite`选项，以指示必须替换单元中的现有表。在将数据导入HDFS或省略此步骤之后，Sqoop将生成一个Hive脚本，其中包含使用Hive的类型定义列的CREATE表操作，并生成`LOAD Data INPATH`语句将数据文件移动到Hive的仓库目录中。

在导入Hive之前先要配置Hadoop的Classpath才可以,否则会报类找不到错误,在`/etc/profile`末尾添加如下配置:

```
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HIVE_HOME/lib/*

#刷新配置
source /etc/profile
```

4.3.4.3.2 参数说明

具体的参数如下:

Argument	Description
`--hive-home`	覆盖环境配置中的`$HIVE_HOME`,默认可以不配置
--hive-import	指定导入数据到Hive中
`--hive-overwrite`	覆盖当前已有的数据
`--create-hive-table`	是否创建hive表,如果已经存在,则会失败
`--hive-table`	设置要导入的Hive中的表名

4.3.4.3.3 实际导入案例

具体导入演示代码如下:

> 提示: 为了看到演示效果,可以先在Hive删除emp表

```
[root@qianfeng01 sqoop-1.4.7] bin/sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table emp \
--hive-import \
--hive-overwrite \
--hive-table "emp3" \
--hive-database db2 \
-m 1
```

在Hive中查看表:

```
hive> show tables;
#结果如下:
OK
emp
```

可以在Hive中查看数据是否导入:

```
select * from emp;
#结果如下:
7369    SMITH   CLERK   7902    1980-12-17      800.0   NULL    20
7499    ALLEN   SALESMAN        7698    1981-02-20      1600.0  300.0   30
7521    WARD    SALESMAN        7698    1981-02-22      1250.0  500.0   30
7566    JONES   MANAGER 7839    1981-04-02      2975.0  NULL    20
7654    MARTIN  SALESMAN        7698    1981-09-28      1250.0  1400.0  30
7698    BLAKE   MANAGER 7839    1981-05-01      2850.0  NULL    30
7782    CLARK   MANAGER 7839    1981-06-09      2450.0  NULL    10
7788    SCOTT   ANALYST 7566    1987-04-19      3000.0  NULL    20
7839    KING    PRESIDENT       NULL    1981-11-17      5000.0  NULL    10
7844    TURNER  SALESMAN        7698    1981-09-08      1500.0  0.0     30
7876    ADAMS   CLERK   7788    1987-05-23      1100.0  NULL    20
7900    JAMES   CLERK   7698    1981-12-03      950.0   NULL    30
7902    FORD    ANALYST 7566    1981-12-03      3000.0  NULL    20
7934    MILLER  CLERK   7782    1982-01-23      1300.0  NULL    10

```

4.4 Sqoop导入实战

4.4.1 Sqoop-import

案例1

表没有主键，需要指定map task的个数为1个才能执行

Sqoop导入原理:

> Sqoop默认是并行的从数据库源导入数据。您可以使用-m或--num-mappers参数指定用于执行导入的map任务(并行进程)的数量。每个参数都取一个整数值，该整数值对应于要使用的并行度。默认情况下，使用四个任务。一些数据库可以通过将这个值增加到8或16来改善性能。
>
> 默认情况下，Sqoop将标识表中的主键id列用作拆分列。从数据库中检索分割列的高值和低值，map任务操作整个范围的大小均匀的组件。譬如ID的范围是0-800,那么Sqoop默认运行4个进程,通过执行 `SELECT MIN(id), MAX(id) FROM emp` 找出id的范围,然后把4个任务的id设置范围是(0-200),(200-400),(400-600),(600-800)
>
> 但是当一个表没有主键时,上面的切分就无法进行,Sqoop导入时就会出错,这时候可以通过-m把mapper的数量设为1,只有一个Mapper在运行,这时候就不需要切分,也可以避免主键不存在时候报错的问题.
>
> ```
> #错误信息
> ERROR tool.ImportTool: Import failed: No primary key could be found for table emp.
> Please specify one with --split-by or perform a sequential import with '-m 1'.
> 
> ```

导入代码:

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp -m 1
```

4.4.2 DBMS-HDFS

案例2

表没有主键，使用--split-by指定执行split的字段

> 问题同上,如果表没有主键,那么还有个办法就是手工指定要拆分的列,通过`--split-by`来指定

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp \
--split-by empno \
--delete-target-dir \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp
```

```
-- 出错
Caused by: java.sql.SQLException: null,  message from server: "Host 'qianfeng01' is not
allowed to connect to this MySQL server"
```

解决方案：

先连接MySql：

```
[root@qianfeng01 sqoop-1.4.7]# mysql -uroot -p
```

(执行下面的语句 .:所有库下的所有表 %：任何IP地址或主机都可以连接)

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'mysql' WITH GRANT OPTION;
    FLUSH PRIVILEGES;
```

案例3:条件导入(增量导入)

需要导入的数据不是全部的，而是带条件导入

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--table emp \
--split-by empno \
--where 'empno > 7777' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/emp
```

案例4:部分字段导入

要导入的数据，不想包含全部字段，只需要部分字段

注意:这种跟where差不多,使用时更灵活一些

```
[root@qianfeng01 sqoop-1.4.7] bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb \
--username root --password 123456 \
--split-by empno \
--query 'select empno,ename,job from emp where empno > 7777 and $CONDITIONS' \
--target-dir hdfs://qianfeng01:8020/sqoopdata/7
```

4.4.3 DBMS-Hive

案例5：将数据导入到Hive中

```
[root@qianfeng01 sqoop-1.4.7]# bin/sqoop import --connect jdbc:mysql://localhost:3306/qfdb
--username root
--password 123456
--table emp
--hive-import
-m 1
```

4.4.4 DBMS-HBase

把数据导入到HBase中

```
hbase中创建表:
create 'mysql2hbase','info'

# 方法一:
[root@qianfeng01 sqoop-1.4.7]# sqoop import  --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table emp \
--hbase-table mysql2hbase \
--column-family info \
--hbase-create-table \
--hbase-row-key empno \
-m 1 \

测试:
hbase(main):008:0> scan 'mysql2hbase'
ROW                                   COLUMN+CELL
 1                                    column=info:hobby, timestamp=1585852383291, value=1
 1                                    column=info:profile, timestamp=1585852383291,
value=\xE6\xBC\x94\xE5\x91\x98
 1                                    column=info:uname, timestamp=1585852383291,
value=bingbing
 2                                    column=info:hobby, timestamp=1585852383291, value=2
 2                                    column=info:profile, timestamp=1585852383291,
value=\xE6\xBC\x94\xE5\x91\x98
 2                                    column=info:uname, timestamp=1585852383291,
value=feifei
 3                                    column=info:hobby, timestamp=1585852383291, value=1
```

```
 3                                          column=info:profile, timestamp=1585852383291,
value=\xE5\x94\xB1\xE6\xAD\x8C
 3                                          column=info:uname, timestamp=1585852383291,
value=\xE5\x8D\x8E\xE4\xBB\x94
3 row(s) in 2.2770 seconds


# 方法二:
hbase(main):004:0> create 'mysql2hbase11','info'
[root@qianfeng01 sqoop-1.4.7]# sqoop import  --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table emp \
--hbase-table mysql2hbase11 \
--delete-target-dir \
--column-family info \
--hbase-create-table \
--hbase-row-key empno \
-m 1 \
--hbase-bulkload

运行后在结尾处有结果(Trying to load hfile):
s20/04/03 10:41:11 WARN mapreduce.LoadIncrementalHFiles: Skipping non-directory
hdfs://qianfeng01:8020/user/root/user_info/_SUCCESS
h20/04/03 10:41:12 INFO hfile.CacheConfig: CacheConfig:disabled
a20/04/03 10:41:12 INFO mapreduce.LoadIncrementalHFiles: Trying to load
hfile=hdfs://qianfeng01:8020/user/root/emp/info/1aef7d02d1a646008f18d49cbb23f20f first=1
last=3


注:
-- hbase-bulkload 不用输入路径，会自己默认导出到某目录，然后完成后自行装载数据到hbase表中;
-m 需要再--hbase-bulkload之前出现

# 测试:
hbase(main):004:0> scan 'mysql2hbase1'
ROW                                        COLUMN+CELL
 1                                         column=info:hobby, timestamp=1585881667767, value=1
 1                                         column=info:profile, timestamp=1585881667767,
value=\xE6\xBC\x94\xE5\x91\x98
 1                                         column=info:uname, timestamp=1585881667767,
value=bingbing
 2                                         column=info:hobby, timestamp=1585881667767, value=2
 2                                         column=info:profile, timestamp=1585881667767,
value=\xE6\xBC\x94\xE5\x91\x98
 2                                         column=info:uname, timestamp=1585881667767,
value=feifei
 3                                         column=info:hobby, timestamp=1585881667767, value=1
 3                                         column=info:profile, timestamp=1585881667767,
value=\xE5\x94\xB1\xE6\xAD\x8C
 3                                         column=info:uname, timestamp=1585881667767,
value=\xE5\x8D\x8E\xE4\xBB\x94
3 row(s) in 0.6170 seconds
```

```
67
```

4.4.5 增量导入数据

4.4.5.1 使用场景

1. 经常被操作不断产生数据的表，建议增量。
2. 当某表基数很大，但是变化很小，也建议增量

4.4.5.2 使用方式

1. query where： 能精确锁定数据范围
2. incremental： 增量，最后记录值来做的

4.4.5.2.1 query where方式

通过查询具体日期的方式进行导入

新建一个脚本文件

```
mysql中的表格:
 CREATE TABLE qfdb.sales_order(
    orderid INT PRIMARY KEY,
    order_date DATE
    )
[root@qianfeng01 sqoop-1.4.7] vi ./import.sh
```

写入以下内容:

```
#!/bin/bash
# yesterday=`date -d "1 days ago" "+%Y-%m-%d"`
yesterday=$1
sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--query "select * from sales_order where DATE(order_date) = '${yesterday}' and \$CONDITIONS"
\
--driver com.mysql.jdbc.Driver \
--delete-target-dir \
--target-dir /user/hive/warehouse/sales_order/dt=${yesterday} \
--split-by id \
-m 1 \
--fields-terminated-by '\t' \
--null-string '\\N' \
--null-non-string '0'
```

```
注:--null-string '\\N'表示字符串的null
--null-non-string '0' 表示非字符串的null
```

执行

```
[root@qianfeng01 sqoop-1.4.7]# bash import.sh 2019-02-01
```

通过下面HDFS可以快速查询到结果:

```
 [root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /user/hive/warehouse/sales_order/dt=2019-01-01/pa*
```

4.4.4.2.2 increment的append方式

```
#将会手动维护last-value
[root@qianfeng01 sqoop-1.4.7]# sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table sales_order \
--driver com.mysql.jdbc.Driver \
--target-dir /user/hive/warehouse/sales_order1/dt=2019-12-30 \
--split-by order_id \
-m 1 \
--check-column order_number \
--incremental append \
--last-value 800 \
--fields-terminated-by '\t' \
--null-string '\\N' \
--null-non-string '0'
```

```
注意:--last-value 80000 \  从80000开始检查,如果后面有新的数据就会进行增量导入,如果没有新的数据会提示下面的信息
21/12/12 01:52:16 INFO tool.ImportTool: Incremental import based on column order_date
21/12/12 01:52:16 INFO tool.ImportTool: No new rows detected since last import.
```

使用下面命令查看:

```
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -cat /user/hive/warehouse/sales_order1/dt=2019-12-30/pa*
```

4.5 Sqoop导出

4.5.1 普通导出

在Sqoop中,使用export进行导出,指的是从HDFS中导出数据到MySql中:

1. 构建MySql的表：

```
CREATE TABLE `u2` (
  `id` int(11) DEFAULT NULL,
  `age` int(11) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `u3` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(20) default NULL,
  `age` int(11) DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2. HDFS导出到MySql

```
第一种：
导入语句
[root@qianfeng01 sqoop-1.4.7]# sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table u22  \
--driver com.mysql.jdbc.Driver \
--target-dir /1906sqoop/u22 \
--split-by id -m 1   \
--fields-terminated-by '\t' \
--null-string '\\N'  \
--null-non-string '0';

导出语句：
[root@qianfeng01 sqoop-1.4.7]# sqoop export --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table u2 \
--driver com.mysql.jdbc.Driver \
--export-dir '/1906sqoop/u22/*' \
--input-fields-terminated-by '\t' \
--input-null-string '\\N' \
--input-null-non-string '\\N' \
-m 1

第二种：

导入语句
[root@qianfeng01 sqoop-1.4.7]# sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--query 'select id,name,age from stu where id > 6 and $CONDITIONS' \
--driver com.mysql.jdbc.Driver \
--delete-target-dir \
--target-dir '/1906sqoop/u7' \
--split-by id \
-m 1 \
```

```
--fields-terminated-by '\t' \
--null-string '\\N' \
--null-non-string '0'


导出语句:
[root@qianfeng01 sqoop-1.4.7]# sqoop export --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table u3 \
--driver com.mysql.jdbc.Driver \
--export-dir '/1906sqoop/u7/*' \
--input-fields-terminated-by '\t' \
--input-null-string '\\N' \
--input-null-non-string '\\N' \
-m 1
```

要注意以下问题

- MySql表的编码格式做为utf8，HDFS文件中的列数类型和MySql表中的字段数一样,最好指定分隔符
- 导出暂不能由Hbase表导出MySql关系型数据库中
- `--export-dir`是一个hdfs中的目录，它不识别_SUCCESS文件
- --query导入的时候注意设置问题。
- 导出数据中有些列值有"null"，会报没法解析
- 导出数据的类型需要和MySql中的一致(能自动转没有问题)

4.5.2 更新并插入导出

场景:

多维结果数据导出；异常重跑数据

```
--update-mode :
updateonly，是默认，仅更新,不会新增数据;
allowinsert : 更新并允许插入
--update-key :  指定更新字段
```

```
CREATE TABLE `upv` (
  `country_id` int(11) NOT NULL AUTO_INCREMENT,
  `visits` int(11) DEFAULT NULL,
  PRIMARY KEY (`country_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

导入数据:
[root@qianfeng01 sqoop-1.4.7]# sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table upv1 \
```

```
--driver com.mysql.jdbc.Driver \
--target-dir /1906sqoop/upv1 \
--split-by country_id -m 1  \
--fields-terminated-by ',' \
--null-string '\\N' \
--null-non-string '0'; \

导出语句:
[root@qianfeng01 sqoop-1.4.7]# sqoop export --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table upv1 \
--export-dir /1906sqoop/upv1/* \
--input-fields-terminated-by "," \
--update-mode allowinsert \
--update-key country_id
```

4.5.3 Sqoop导出parquet格式的数据

导入数据到HDFS中为parquet格式:

```
[root@qianfeng01 sqoop-1.4.7]# sqoop import --connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--query 'select id,age from u2 where id > 2 and $CONDITIONS' \
--driver com.mysql.jdbc.Driver \
--delete-target-dir \
--target-dir '/1906sqoop/u8' \
--split-by id \
-m 1 \
--fields-terminated-by '\t' \
--null-string '\\N' \
--null-non-string '0' \
--as-parquetfile
```

导出语句:

```
# 创建表:
CREATE TABLE `par` (
  `id` int(11) NOT NULL DEFAULT '0',
  `age` int(1) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

# hive创建表:
hive> create  table if not exists par2(
`id` int,
 `age` int
)
row format delimited fields terminated by '\t'
stored as parquet
```

```
location '/1906sqoop/u8/'
;
```

将hive包中的lib目录下的hcatalog相关包拷贝到sqoop的lib目录中去：

```
考虑将hcatalog的包放到sqoop中或者做环境变量。
必须保证hive对mysql有执行权限
grant all privileges on *.* to 'root'@'%' identified by '123456' with grant option;
```

导出parquet格式语句：

```
[root@qianfeng01 sqoop-1.4.7] sqoop export \
--connect jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table par \
--hcatalog-database default \
--hcatalog-table par2 \
-m 1

参数说明：
--table：MySQL库中的表名
--hcatalog-database：Hive中的库名
--hcatalog-table：Hive库中的表名，需要抽数的表
```

4.6 Sqoop的Job

4.6.1 Job操作

```
job的好处：
1、一次创建，后面不需要创建，可重复执行job即可
2、它可以帮我们记录增量导入数据的最后记录值
3、job的元数据默认存储目录：$HOME/.sqoop/
4、job的元数据也可以存储于mysql中。
```

Sqoop提供一系列的Job语句来操作Sqoop。

```
$ sqoop job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]
$ sqoop-job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]
```

使用方法：

```
usage: sqoop job [GENERIC-ARGS] [JOB-ARGS] [-- [<tool-name>] [TOOL-ARGS]]

Job management arguments:
   --create <job-id>            Create a new saved job
   --delete <job-id>            Delete a saved job
   --exec <job-id>              Run a saved job
   --help                       Print usage instructions
   --list                       List saved jobs
   --meta-connect <jdbc-uri>    Specify JDBC connect string for the
                                metastore
   --show <job-id>              Show the parameters for a saved job
   --verbose                    Print more information while working

```

列出Sqoop的Job：

```
[root@qianfeng01 sqoop-1.4.7] sqoop job --list
```

创建一个Sqoop的Job：

```
[root@qianfeng01 sqoop-1.4.7]# sqoop job --create sq2 -- import  --connect
jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table u2 \
--driver com.mysql.jdbc.Driver \
--delete-target-dir \
--target-dir '/1906sqoop/u10' \
--split-by id \
-m 1

注意:第一行的--与import之间有空格
```

执行Sqoop的Job：

```
#如报错json包找不到，则需要手动添加
sqoop job --exec sq1


执行的时候会让输入密码：
输入该节点用户的对应的密码即可
# 1、配置客户端记住密码(sqoop-site.xml)追加
 <property>
    <name>sqoop.metastore.client.record.password</name>
    <value>true</value>
  </property>

# 2、将密码配置到hdfs的某个文件，我们指向该密码文件
```

```
说明：在创建Job时，使用--password-file参数，而且非--passoword。主要原因是在执行Job时使用--password参数将有警告，并且需要输入密码才能执行Job。当我们采用--password-file参数时，执行Job无需输入数据库密码。
[root@qianfeng01 sqoop-1.4.7]# echo -n "root" > sqoop.pwd
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -put sqoop.pwd /input
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -chmod 400 /input/sqoop.pwd
[root@qianfeng01 sqoop-1.4.7]# hdfs dfs -ls /input
-r-------- 1 hadoop supergroup 6 2018-01-15 18:38 /input/sqoop.pwd
```

查看Sqoop的Job:

```
[root@qianfeng01 sqoop-1.4.7] sqoop job --show sq1
```

删除Sqoop的Job:

```
[root@qianfeng01 sqoop-1.4.7] sqoop job --delete sq1
```

问题:

```
1 创建job报错：19/12/02 23:29:17 ERROR sqoop.Sqoop: Got exception running Sqoop: java.lang.NullPointerException
java.lang.NullPointerException
        at org.json.JSONObject.<init>(JSONObject.java:144)

解决办法：
添加java-json.jar包到sqoop的lib目录中。
如果上述办法没有办法解决，请注意hcatlog的版本是否过高，过高将其hcatlog包剔除sqoop的lib目录即可。

2 报错：Caused by: java.lang.ClassNotFoundException: org.json.JSONObject
解决办法：
添加java-json.jar包到sqoop的lib目录中。
```

4.6.2 metastore服务

```
metastore服务是元数据服务，用于存储sqoop的job相关信息，将信息保存于关系型数据库中.默认的元数据库在当前用户目录下的.sqoop目录中.
```

优点：

```
1 job信息更加有保障
2 多个client之间共享job信息。
```

服务器规划情况：

```
qianfeng01 metastore、client(不用启动服务)
qianfeng02 client(不用启动服务)
```

qianfeng01的metastore的配置：

注意报类似如下的错误：ERROR org.apache.sqoop.metastore.hsqldb.HsqldbJobStorage - Cannot restore job。

（1）在MySql中创建Sqoop的元数据存储数据库 （如果有root可以用root）

这里默认要求设置的密码中等强度安全:大小写+数字+特殊字符

```
create database sqoop;
create user 'sqoop'@'%' identified by 'Sqoop123#';
grant all privileges on sqoop.* to 'sqoop'@'%';
flush privileges;
```

（2）配置Sqoop的元数据存储参数 在`$SQOOP_HOME/conf/sqoop-site.xml`中添加以下的参数,在`/code/sqoop-site.xml`也可以查阅。

```
sqoop.metastore.server.location：指定元数据服务器位置，初始化建表时需要。
sqoop.metastore.client.autoconnect.url：客户端自动连接的数据库的URL。
sqoop.metastore.client.autoconnect.username：连接数据库的用户名。
sqoop.metastore.client.enable.autoconnect：启用客户端自动连接数据库。
sqoop.metastore.client.record.password：在数据库中保存密码，不需要密码即可执行sqoop job脚本。
sqoop.metastore.client.autoconnect.password：连接数据库的密码。

  <property>
    <name>sqoop.metastore.client.enable.autoconnect</name>
    <value>false</value>
    <description>If true, Sqoop will connect to a local metastore
      for job management when no other metastore arguments are
      provided.
    </description>
  </property>

  <property>
    <name>sqoop.metastore.client.autoconnect.url</name>
    <value>jdbc:mysql://qianfeng01:3306/sqoop</value>
  </property>
  <property>
    <name>sqoop.metastore.client.autoconnect.username</name>
    <value>root</value>
  </property>
  <property>
    <name>sqoop.metastore.client.autoconnect.password</name>
    <value>123456</value>
  </property>

  <property>
    <name>sqoop.metastore.client.record.password</name>
    <value>true</value>
  </property>

```

```
  <property>
    <name>sqoop.metastore.server.location</name>
    <value>/usr/local/sqoop/sqoop-metastore/shared.db</value>
  </property>

  <property>
    <name>sqoop.metastore.server.port</name>
    <value>16000</value>
  </property>
```

（3）重启Sqoop服务 保存配置并重启完成后，MySql的Sqoop库中有了一个名为SQOOP_ROOT的空表。

注意:这个表开始是没有的,我们需要执行一些操作,这个表才会出来,比如:

sqoop job --list --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456'

这里会报错,不用管,去查看sqoop库下面SQOOP_ROOT表已经出现了.

```
#启动:
[root@qianfeng01 sqoop-1.4.7]# sqoop metastore &

#查看进程:
[root@qianfeng01 sqoop-1.4.7]# jps
sqoop

#关闭:
[root@qianfeng01 sqoop-1.4.7]# sqoop metastore --shutdown
```

（4）预装载S表

```
insert into SQOOP_ROOT values (NULL, 'sqoop.hsqldb.job.storage.version', '0');
```

（5）Job相关操作

```
[root@qianfeng01 sqoop-1.4.7]# sqoop job --list  ###需要加--meta-connect

创建job:
[root@qianfeng01 sqoop-1.4.7]# sqoop job --create sq3 --meta-connect
'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' -- import  --connect
jdbc:mysql://qianfeng01:3306/qfdb \
--username root \
--password 123456 \
--table u2 \
--driver com.mysql.jdbc.Driver \
--delete-target-dir \
--target-dir '/1906sqoop/u10' \
--split-by id \
-m 1

列出job:
```

```
[root@qianfeng01 sqoop-1.4.7]# sqoop job --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' --list

执行job:
[root@qianfeng01 sqoop-1.4.7]# sqoop job --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' --exec sq3

执行job并打印详细信息:
[root@qianfeng01 sqoop-1.4.7]# sqoop job --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' --exec sq3 -verbose
```

此时并不会返回先前已经创建的myjob_incremental_import作业，因为此时MySql中没有元数据信息。该命令执行完成后，MySql的Sqoop库中有了一个名为SQOOP_SESSIONS的空表，该表存储Sqoop Job相关信息。

（6）将表的存储引擎修改为MYISAM(如job信息存储到MySql的SQOOP_SESSIONS则不用执行如下)

```
alter table SQOOP_ROOT engine=myisam;
alter table SQOOP_SESSIONS engine=myisam;
```

因为每次执行增量抽取后都会更新last_value值，如果使用Innodb可能引起事务锁超时错误。

（7） qianfeng02的配置就是把qianfeng01的拷贝过去即可

```
[root@qianfeng01 sqoop-1.4.7]# rm -rf ./docs/
[root@qianfeng01 sqoop-1.4.7]# scp -r ../sqoop/ qianfeng02:/usr/local/

#qianfeng02上操作如下:
[root@qianfeng02 sqoop-1.4.7]# ./bin/sqoop job --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' --list

执行job:
[root@qianfeng02 sqoop-1.4.7]# ./bin/sqoop job --meta-connect 'jdbc:mysql://qianfeng01:3306/sqoop?user=root&password=123456' --exec sq3
```

4.7 Sqoop优化

4.7.1 -m与split-by的优化

1. 小量数据时(200M左右) :最好使用一一个map，快且减少小文件。
2. 大量数据时:要特别考虑数据的特征，对于split- by最完美的情况是有一个:均匀分布的数字(如自增列)或时间字段，且这个字段还有索引(最好字段是int、tinyin)，这样在抽取时使得并发的每个sq1处理相近的数据量，并且Sqoop附加的where条件可以使用索引。
3. split-by id，-m 2， 数据量1-100。第 一个mapper:(0,50]第二个mapper: (50, 100],对于m要综合考虑数据量、I0、源数据库的性能、集群的资源等等。一种简单的考虑是最大不超过yarn.上分配给这个用户的core个数，最小"数据量/m"要够一个128MB的文件。如果条件允许可以先设置一个值跑着试试，然后观察源数据库负载、集群I0以及运行时长等，再进行相应调整。

4.7.2 --fetch-size n

一次取MySql中批量读取的数据条数。建议优化如下:

1. 考虑一条数据的量。(如果2个字段和200个字段的--fetch-size不能一样)
2. 考虑数据库的性能
3. 考虑网络速度
4. 最好的状态是一 次--fetch-size能满足一个mapper

五 实战应用

六 教学总结

6.1 教学重点

```
Sqoop和HDFS导入导出
Sqoop和Hive的导入导出
Sqoop和HBase的导出
```

6.2 教学难点

```
Sqoop和HBase的导出
Sqoop的Job分析
Sqoop优化
```

七 课后作业

习题

第一题

```
请简述Sqoop的架构和安装过程
```

第二题

用sqoop实现数据库连接,,完成下面功能:

```
1.列出所有数据库
2.列出数据库中所有表的sql,
3.命令行参数执行
4.用脚本文件两种方式执行.
```

第三题

用sqoop实现数据库表的下面三种导入方式

```
1.指定列导入
2.查询条件导入
3.Sql导入等三种方式.
```

第四题

用sqoop实现导入数据,要分别解决下面几种问题,

```
1.没有主键
2.增量插入
```

第五题

用脚本实现Sqoop的从HDFS导出到Mysql

八 解决方案

8.1 应用场景

8.2 核心面试题