sqoop 常用命令

# 一、Sqoop与MySQL

## 1.列出mysql数据库中的所有数据库

sqoop list-databases -connect jdbc:mysql://192.168.10.111:3306 -username root -password root

**注意：**

以下URL写法，都可以

jdbc:mysql://192.168.1.10:3306/（推荐）

jdbc:mysql://192.168.1.10/

jdbc:mysql://192.168.1.10

jdbc:mysql://masters:3306/

jdbc:mysql://master/

jdbc:mysql://master

jdbc:mysql://localhost:3306/

jdbc:mysql://localhost/

jdbc:mysql:///

jdbc:mysql://

## 2.列出数据库sqoop中的所有数据表(列出mysql中sqoop这个数据库所有表)

sqoop list-tables -connect jdbc:mysql:///sqoop -username root -password root

## 3.通过Sqoop执行SQL语句

sqoop eval -connect jdbc:mysql:///sqoop -username root -password root -query ‘select \* from emp $CONDITIONS’

可以快速地使用SQL语句对关系数据库进行操作，这可以使得在使用import这种工具进行数据导入的时候，可以预先了解相关的SQL语句是否正确，并能将结果显示在控制台。

# 二、Sqoop与HDFS

## 1.将sqoop.emp表中的数据导入HDFS的/sqfs目录下

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -m 1 -target-dir /sqfs/22

**SQL语句**

（1）、有where条件：

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -query 'select \* from emp where eNo=7782 and $CONDITIONS' -m 2 --split-by eNo -target-dir /sqfs/33;

（2）、没有where条件：

sqoop import --connect jdbc:mysql:///sqoop --username root --password root --query 'select \* from emp where $CONDITIONS' --split-by eNo -m 1 --target-dir /sqfs/44

注意:

Cannot specify --query and --table together.

-query 不能和 -table在一起搭配使用

-query 查询语句中必须要有 where 和 $CONDITIONS(来结束)

-query 查询语句中where 后不能有条件

## 2.将HDFS上/sqfs目录下的数据导入的sqoop.emp2表中

sqoop export -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp2 -m 1 -export-dir /sqfs/22

注意：必须在关系型数据中创建emp2表。

# 三、Sqoop与Hive

## 1.将关系型数据的emp表结构复制到Hive中

sqoop create-hive-table -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -hive-table emp -fields-terminated-by "\0001" -lines-terminated-by "\n"

注：

-hive-table emp指定在Hive中创建的表名为emp（默认数据库default）

-hive-table hsqoop.emp指定在Hive中的hsqoop数据库下创建emp表

-fields-terminated-by "\0001"  是设置每列之间的分隔符，"\0001"是ASCII码中的1，是hive的默认行内分隔符，而sqoop的默认行内分隔符为"\0001"

-lines-terminated-by "\n"  设置的是每行之间的分隔符，此处为换行符，也是默认的分隔符；

## 2.将关系数据库中的emp表的数据导入文件到Hive表中

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -hive-table emp -m 1 -fields-terminated-by "\0001" -hive-import

注：

-fields-terminated-by "\0001" 需同创建Hive表时保持一致

-hive-import 指定是Hive导入数据

-split-by id emp中没有主键时，用于指定Mapper时的Key

## 3. Hive导入参数

-hive-home <dir> 重写$HIVE\_HOME

-hive-import 插入数据到hive当中，使用hive的默认分隔符

-hive-overwrite 重写插入

-create-hive-table 建表，如果表已经存在，该操作会报错！

-hive-table <table-name> 设置到hive当中的表名

-hive-drop-import-delims 导入到hive时删除 \n, \r, and \0001

-hive-delims-replacement 导入到hive时用自定义的字符替换掉 \n, \r, and \0001

-hive-partition-key hive分区的key

-hive-partition-value <v> hive分区的值

-map-column-hive <map> 类型匹配，sql类型对应到hive类型

**hive空值处理**

sqoop会自动把NULL转换为null处理，但是hive中默认是把\N来表示null，因为预先处理不会生效的，我们需要使用 -null-string 和 -null-non-string来处理空值 把\N转为[\\N](file://n/)

例句：sqoop import ... -null-string '\\N' 或-null-non-string '\\N'

**sqoop导入hive数据到MySql碰到hive表中列的值为null的情况：**

在导入数据的过程中，如果碰到列值为null的情况，hive中为null的是以\N代替的，所以你在导入到MySql时，需要加上两个参数：--input-null-string '\\N' --input-null-non-string '\\N'，多加一个'\'，是为转义。如果你通过这个还不能解决字段为null的情况，还是报什么NumberFormalt异常的话，那就是比较另类的了，没有关系，我们还是要办法解决。

你应该注意到每次通过sqoop导入MySql数据时，都会生成一个以MySql表命名的.java文件，然后打成JAR包，给sqoop提交给hadoop 的MR来解析Hive表中的数据。那我们可以根据报的错误，找到对应的行，改写该文件，编译，重新打包，sqoop可以通过 -jar-file ，--class-name 组合让我们指定运行自己的jar包中的某个class。来解析该hive表中的每行数据。脚本如下：一个完整的例子如下：

sqoop export --connect

"jdbc:mysql://localhost/aaa?useUnicode=true&characterEncoding=utf-8"

--username aaa --password bbb --table table

--export-dir /hive/warehouse/table --input-fields-terminated-by '\t'

--input-null-string '\\N' --input-null-non-string '\\N'

--class-name com.chamago.sqoop.codegen.bi\_weekly\_sales\_item

--jar-file /tmp/sqoop-chamago/bi\_weekly\_sales\_item.jar

上面--jar-file 参数指定jar包的路径。--class-name 指定jar包中的class。  
这样就可以解决所有解析异常了。

## 4.将Hive中的表数据导入到mysql数据库emp3表中

sqoop export -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp3 -export-dir /user/hive/warehouse/emp/part-m-00000 -input-fields-terminated-by '\0001'

注：

**在进行导入之前，mysql中sqoop数据库中emp3表必须已经提起创建好了。**

jdbc:mysql://192.168.1.10:3306/sqoop中的IP地址改成localhost会报异常

指定/user/hive/warehouse/sqoop.db/emp/part-m-00000，只加载该文件

指定/user/hive/warehouse/sqoop.db/emp/，加载该目录下的所有文件

-input-fields-terminated-by '\0001' :hive表创建时的分隔符为’\0001’

# 四、Sqoop与->HBase

## 1. MySQL 中的emp表中的数据导入数据到 HBase的emp表中

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -hbase-table emp -column-family 'femp' -hbase-row-key eNo -m 1

注：

-hbase-table emp 指定HBase的表emp

-column-family 'femp' 指定列族名femp

-hbase-create-table 该参数是用来创建HBase表的，但不太管用

**注意：因为版本的问题可能会出现比较多的问题：**

**1、sqoop-env.sh 配置好zookeeper**

**2、hbase每个节点都要能访问mysql数据库**

**3、因为版本的问题的话必须先在hbase先创建库：create ‘emp’,’femp’**

**4、顺序执行，指定eNo作为行键，其它字段都顺序添加到femp**

## 2. 将Mysql数据库emp表不同列的数据添加到emp3表中不同列族中

* 先将emp表的eName,eJob列中的数据到f1\_emp列族中

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -hbase-table emp3 -column-family 'f1\_emp' -hbase-row-key eNo -m 1 -columns eNo, eName,eJob

注：

-columns id,name 指定emp表中的eName,eJob列

-hbase-row-key id 指定emp表中的行eNo

* 先将emp表的emgr,eHiredate,eSal,ecomm,dp\_no列中的数据到f2\_emp列族中

sqoop import -connect jdbc:mysql://192.168.10.111:3306/sqoop -username root -password root -table emp -hbase-table emp3 -column-family 'f2\_emp' -hbase-row-key eNo -m 1 -columns eNo,emgr,eHiredate,eSal,ecomm,dp\_no

**注意：先创建两个列族：create 'emp3','f1\_emp','f2\_emp'**

# 五、重要参数

## 1.转换为对象

-map-column-java <mapping>  将转换为java数据类型

-map-column-hive <mapping>  将转换为hive数据类型

## 分隔符、转义字符

例句：

Some string, with a comma.

Another "string with quotes"

导入命令：

$ sqoop import -fields-terminated-by , -escaped-by \\ -enclosed-by '\"' ...

处理结果：

"Some string, with a comma.","1","2","3"...

"Another \"string with quotes\"","4","5","6"...

导入命令：

$ sqoop import -optionally-enclosed-by '\"' (the rest as above)...

处理结果：

"Some string, with a comma.",1,2,3...

"Another \"string with quotes\"",4,5,6...

# 六、常用工具

## 1. sqoop job

保存常用的作业，以便下次快速调用

-create <job-id> 创建一个新的job  
　　-delete <job-id> 删除job  
　　-exec <job-id>  执行job  
　　-show <job-id> 显示job的参数  
　　-list 列出所有的job

* 创建job

sqoop job -create myjob - import -connect jdbc:mysql://example.com/db ... -table mytable

* 列出所有job

sqoop job -list

* 查看job

sqoop job -show myjob

* 执行job

sqoop job -exec myjob

* 重写参数

sqoop job -exec myjob -username someuser -P

## 2. 聚合工具

sqoop-metastore、sqoop-merge  
合并两个目录

sqoop merge -new-data newer -onto older -target-dir merged -jar-file datatypes.jar -class-name Foo -merge-key id

## 3. 校验Validate

它用来比较源数据和目标数据的数量。

sqoop import --connect jdbc:mysql://db.foo.com/corp ... -table EMPLOYEES --validate

sqoop export --connect jdbc:mysql://db.example.com/foo --table bar -export-dir /results/bar\_data --validate

注：

它有三个接口

**Validator**

Property: validator

Description: Driver for validation,

must implement org.apache.sqoop.validation.Validator

Supported values: The value has to be a fully qualified class name.

Default value: org.apache.sqoop.validation.RowCountValidator

**Validation Threshold**

Property: validation-threshold

Description: Drives the decision based on the validation meeting the

threshold or not. Must implement

org.apache.sqoop.validation.ValidationThreshold

Supported values: The value has to be a fully qualified class name.

Default value: org.apache.sqoop.validation.AbsoluteValidationThreshold

**Validation Failure Handler**

Property: validation-failurehandler

Description: Responsible for handling failures, must implement

org.apache.sqoop.validation.ValidationFailureHandler

Supported values: The value has to be a fully qualified class name.

Default value: org.apache.sqoop.validation.LogOnFailureHandler

例句：

sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES

添加参数：

-validate -validator org.apache.sqoop.validation.RowCountValidator

或-validate -validation-threshold

或-validate -validation-failurehandler

# 配置文件

通过配置文件conf/sqoop-site.xml来配置常用参数

例：

<property>

<name>property.name</name>

<value>property.value</value>

</property>

如果不在这里面配置的话，就需要像这样写命令

sqoop import -D property.name=property.value ...

参数：sqoop.bigdecimal.format.string

作用：大decimal是否保存为string，如果保存为string就是 0.0000007,否则为1E7

参数：sqoop.hbase.add.row.key

作用：是否把作为rowkey的列也加到行数据当中，默认是false的