sqoop import \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--target-dir /junior/sqoop/ \ //可选，不指定目录，数据默认导入到/user下

--where "sex='female'" \ //可选

--as-sequencefile \ //可选，不指定格式，数据格式默认为 Text 文本格式

--num-mappers 10 \ //可选，这个数值不宜太大

--null-string '\\N' \ //可选

--null-non-string '\\N' \ //可选

  2)--password-file：指定一个密码保存文件，读取密码。我们可以将这个文件设置为只有自己可读的文件，防止密码泄露。

sqoop import \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--table user \

--password-file my-sqoop-password

1、Sqoop export 将数据导入数据库，一般情况下是一条一条导入的，这样导入的效率非常低。这时我们可以使用 Sqoop export 的批量导入提高效率，其具体语法如下。

sqoop export \

--Dsqoop.export.records.per.statement=10 \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--export-dir user \

--batch

--Dsqoop.export.records.per.statement：指定每次导入10条数据，--batch：指定是批量导入。

2、在实际应用中还存在这样一个问题，比如导入数据的时候，Map Task 执行失败， 那么该 Map 任务会转移到另外一个节点执行重新运行，这时候之前导入的数据又要重新导入一份，造成数据重复导入。 因为 Map Task 没有回滚策略，一旦运行失败，已经导入数据库中的数据就无法恢复。Sqoop export 提供了一种机制能保证原子性， 使用--staging-table 选项指定临时导入的表。Sqoop export 导出数据的时候会分为两步：第一步，将数据导入数据库中的临时表，如果导入期间 Map Task 失败，会删除临时表数据重新导入；第二步，确认所有 Map Task 任务成功后，会将临时表名称为指定的表名称。

sqoop export \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--staging-table staging\_user

   1)通过 --update-key id 更新已有数据。

sqoop export \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--update-key id

  　　2)使用 --update-key id和--update-mode allowinsert 两个选项的情况下，如果数据已经存在，则更新数据，如果数据不存在，则插入新数据记录。

sqoop export \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--update-key id \

--update-mode allowinsert

 4、如果 HDFS 中的数据量比较大，很多字段并不需要，我们可以使用 --columns 来指定插入某几列数据。

sqoop export \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--column username,sex

 5、当导入的字段数据不存在或者为null的时候，我们使用--input-null-string和--input-null-non-string 来处理。

sqoop export \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

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

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

　2、Sqoop与HBase结合稍微麻烦一些，需要使用 --hbase-table 指定表名称，使用 --column-family 指定列名称。

sqoop import \

--connect jdbc:mysql://db.dajiangtai.net:3306/djtdb\_hadoop \

--username sqoop \

--password sqoop \

--table user \

--hbase-table user \

--column-family city

指定字段类型（pg数据库默认只导public下面的表，可用--schema指定custom模式）

$ sqoop import ... --map-column-java id=String,value=Integer

Notice that specifying commas in --map-column-hive option, you should use URL encoded keys and values, for example, use DECIMAL(1%2C%201) instead of DECIMAL(1, 1).

sqoop import --connect jdbc:postgresql://XXX.XX.XXX.XXX:5432/iAtlas --table msg02 --username aaaa.bbbbb --password ccccc **--schema tracking --map-column-java wsresp=String --map-column-hive wsresp=STRING**

**删除分隔符**

--hive-drop-import-delims Drops \n, \r, and \01 from string fields when importing to Hive.

测试执行

sqoop eval \

--connect jdbc:oracle:thin:@//10.150.33.225:1521/inputndb \

--username TXBI\_READONLY \

--password 1qazXSW@ \

--query "select count(\*) from INPUTNER.HR\_LOG"

**执行如下的指令，实现主键增量的导入Hive**

sqoop import \ --connect jdbc:mysql://master:3306/test \--username hive \ --password 123456 \--table customer \--check-column id \--incremental append \--last-value 5

**Lastmodified导入**

create table customertest(id int,name varchar(20),last\_mod timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP);

**使用incremental的方式进行增量的导入**

sqoop import \ --connect jdbc:mysql://master:3306/test \--username hive \--password 123456 \--table customertest \--check-column last\_mod \--incremental lastmodified \--last-value "2016-12-15 15:47:29" \-m 1 \--append

使用lastmodified模式进行增量处理要指定增量数据是以append模式(附加)还是merge-key(合并)模式添加--merge-key id

**1) 复制MySQL的表结构到Hive**

sqoop create-hive-table --connect jdbc:mysql://192.168.1.1:3306/radius --username root --password 123 --table um\_appuser --hive-table um\_appuser --fields-terminated-by "\0001" --lines-terminated-by "\n"

**2) 创建Hive表对应的目录**

hdfs dfs -mkdir /user/hive/um\_appuser

**3) 修改表Hive表对应的目录**

ALTER TABLE um\_appuser SET LOCATION 'hdfs://node1:8020/user/hive/um\_appuser';

**4) 转换为外部表**

ALTER TABLE um\_appuser SET TBLPROPERTIES ('EXTERNAL'='TRUE');

**将数据导入Sqoop**

sqoop import --connect jdbc:mysql://192.168.1.1:3306/radius --username root --password 123 --table um\_appuser --hive-import --hive-table um\_appuser

**在Hive中执行**

SELECT id FROM um\_appuser ORDER BY id DESC LIMIT 1;

查询结果：15902

**增量导入**

**1) 创建job（注意–last-value的值，上一步查询得到的结果）**

sqoop job --create um\_appuser

-- import

--connect jdbc:mysql://192.168.1.1:3306/radius

--username root --password 123

--table um\_appuser

--hive-import

--hive-table um\_appuser

--incremental append

--check-column id

--last-value 15902

**2) 查看已经创建的Sqoop job**

sqoop job --list

**3) 创建调度任务**

0 \*/1 \* \* \* sqoop job --exec um\_appuser > um\_appuser\_sqoop.log 2>&1 &