

SQOOP 简介

学生指南

商标产权声明

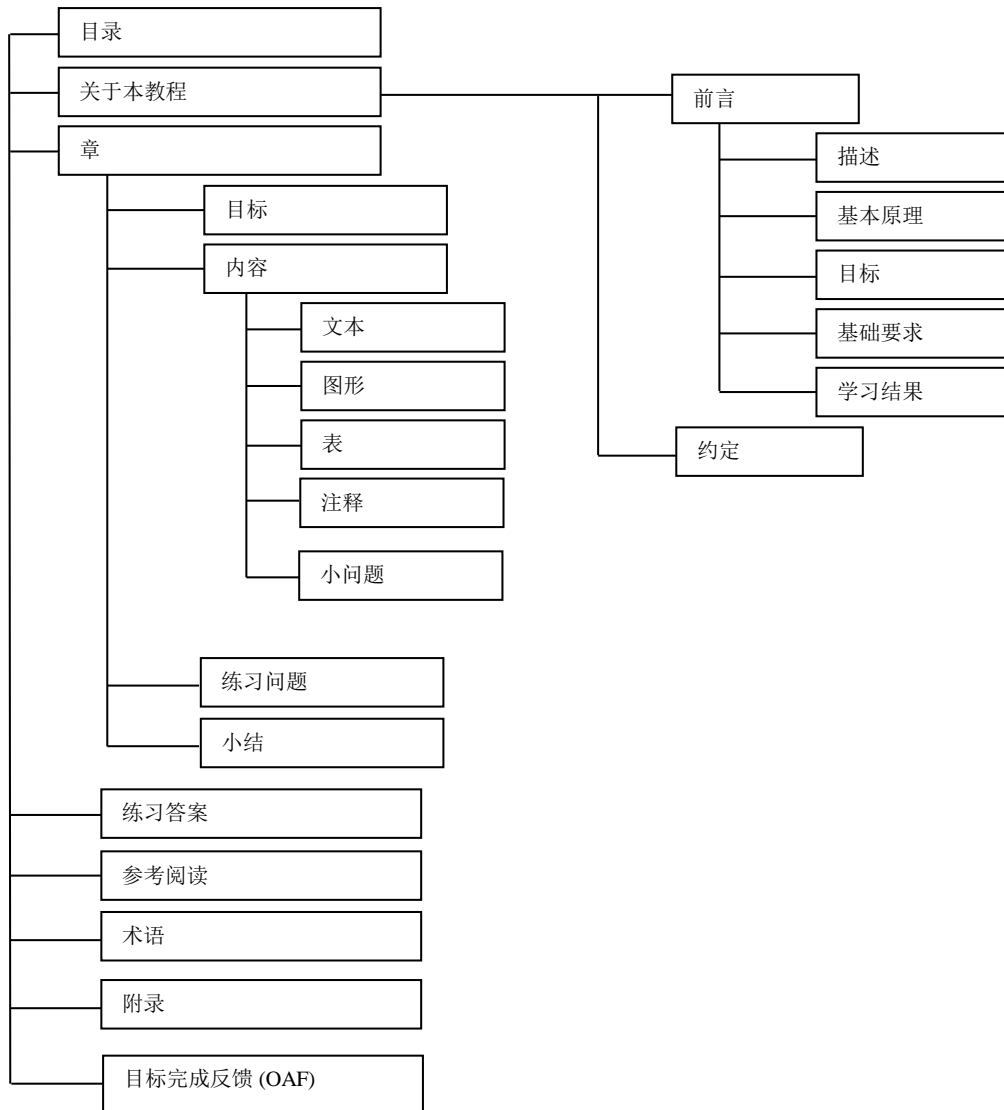
所有产品都是各自组织的注册商标。
所有软件仅用于教育用途。

sqoop 简介 SG/13-M04-V1.0
版权 ©NIIT。保留一切权利。

未经出版商提前书面许可，禁止以任何形式或任何手段（电子、机械、影印、转录或其他方式）对本出版物的某一部分进行复制、传播或将其存储在检索系统中。

印制：Sona Printers Pvt. Ltd. B-181 Okhla Ph-1, N.D.-20 电话：26811313-4-5-6

教程设计 - 学生指南



目录

关于本教程

前言-----	i
描述-----	i
目标-----	i
基础要求-----	ii
学习结果-----	ii
约定-----	iii

第 1 章 - SQOOP 基础

Sqoop 概念-----	1.2
Sqoop 是怎样工作的-----	1.2
Sqoop 安装-----	1.3
数据导入-----	1.5
导入到 hive 中-----	1.11
数据导出-----	1.12
Sqoop 常用命令-----	1.15
Sqoop 高级-----	1.17
练习问题-----	1.9
小结-----	1.10

参考阅读

sqoop 简介-----	R.3
---------------	-----

关于本教程

前言

描述

Sqoop 是一款开源的工具，主要用于在 Hadoop(Hive)与传统的数据库(mysql、postgresql...)间进行数据的传递，可以将一个关系型数据库（例如：MySQL,Oracle,Postgres 等）中的数据导进到 Hadoop 的 HDFS 中，也可以将 HDFS 的数据导进到关系型数据库中。。

目标

完成此教程后，学生将能够熟练使用以下工具：

- Sqoop 安装
- 数据导入
- 数据导出

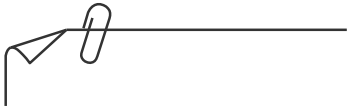

基础要求

想要学习此教程的学生应该具备逻辑构建和有效问题解决的基本知识。

学习结果

完成此教程后，学生将能够使用 相关工具配合 `hadoop`、`hbase`、`storm` 一起使用

约定

约定	表示...
	注释
	小问题
	活动的占位符

SQOOP 基础

Hadoop 中的数据怎么写入关系型数据库中，关系型数据库的数据怎么导入到 hadoop 中。如何在 hadoop 和关系型数据库中建立一个连接点，sqoop 的出现解决了这个问题。

目标

在本章中，您将学习：

- ☞ Sqoop 概念
- ☞ Sqoop 安装
- ☞ Sqoop 导入
- ☞ Sqoop 导出
- ☞ Sqoop 工作原理
- ☞ Sqoop 优化

Sqoop 概念

Sqoop 是 Hadoop 和关系数据库服务器之间传送数据的一种工具。它是用来从关系数据库如：MySQL，Oracle 到 Hadoop 的 HDFS，并从 Hadoop 的文件系统导出数据到关系数据库。

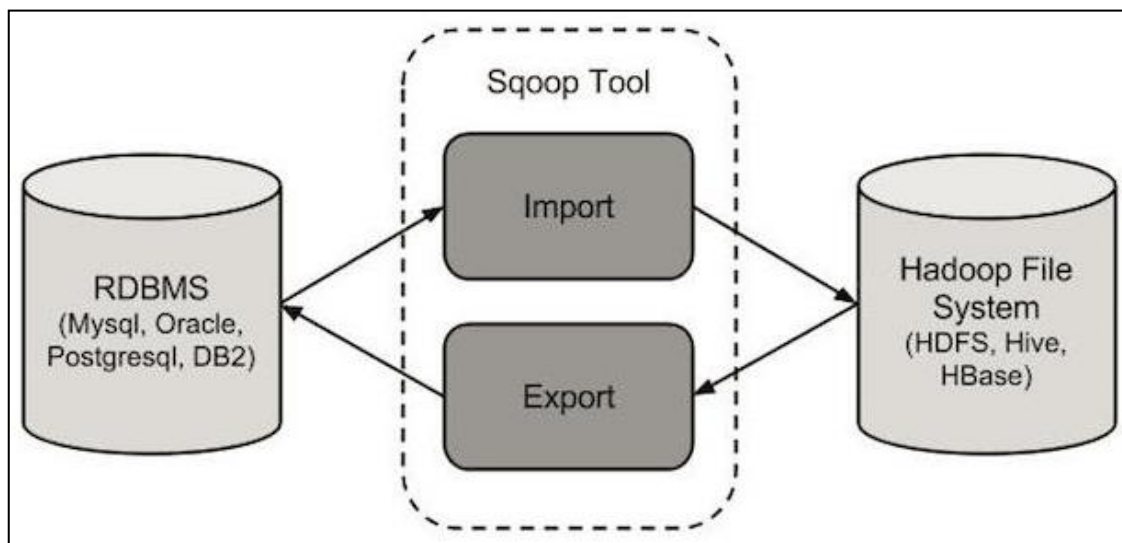
传统的应用管理系统，也就是与关系型数据库的使用 RDBMS 应用程序的交互，是产生大数据的来源之一。这样大的数据，由关系数据库生成的，存储在关系数据库结构关系数据库服务器。

当大数据存储器和分析器，如 MapReduce，Hive，HBase，Cassandra，Pig 等，Hadoop 的生态系统等应运而生图片，它们需要一个工具来用的导入和导出的大数据驻留在其中的关系型数据库服务器进行交互。在这里，Sqoop 占据着 Hadoop 生态系统提供关系数据库服务器和 Hadoop HDFS 之间的可行的互动。

Sqoop: “SQL 到 Hadoop 和 Hadoop 到 SQL”

Sqoop 是怎样工作的？

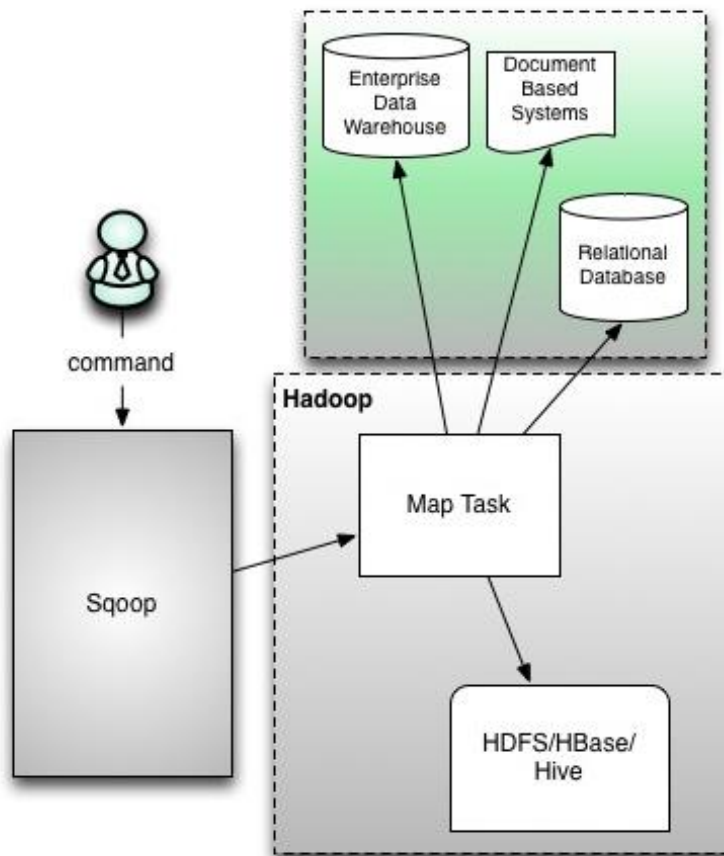
下图描述了 Sqoop 的工作流程



Sqoop 工作流程

Sqoop Tool 提供了 Import 和 Export 工具将关系型数据库中的数据导入到 hadoop 中，例如直接将数据导入 hdfs、hive 或者 hbase 里面。

■ Sqoop 工作原理



Sqoop 解析 SQL 语句后，转变为 MapReduce 后提交到 Hadoop 集群中，然后将数据写入 HDFS

■ Sqoop 导入

导入工具从 RDBMS 到 HDFS 导入单个表。表中的每一行被视为 HDFS 的记录。所有记录被存储在文本文件的文本数据或者在 Avro 和序列文件的二进制数据。

■ Sqoop 导出

导出工具从 HDFS 导出一组文件到一个 RDBMS。作为输入到 Sqoop 文件包含记录，这被称为在表中的行。那些被读取并解析成一组记录和分隔使用用户指定的分隔符。

Sqoop 安装

■ 前提条件:

已经安装了 JDK

已经安装了 hadoop

安装 sqoop

在官网下载 sqoop 的版本 <http://mirror.bit.edu.cn/apache/sqoop/1.4.6/>，选择 sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz，并将安装文件拷贝到/opt/niit/tools 目录下

解压文件

```
tar -zxvf sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz
```

■ 配置环境变量

```
vi /etc/profile
```

增加以下行到文件底部

```
export SQOOP_HOME=/home/soft/sqoop-1.4.6.bin__hadoop-2.0.4-alpha
```

```
export PATH=$PATH:$SQOOP_HOME/bin
```

执行下列命令是环境变量生效

```
source /etc/profile
```

■ 配置 sqoop

要配置 Sqoop 用 Hadoop，需要编辑 sqoop-env.sh 文件，该文件被放置在 \$SQOOP_HOME/conf 目录，执行如下命令

```
cd $SQOOP_HOME/conf
```

```
cp sqoop-env-template.sh sqoop-env.sh
```

找到 HADOOP_COMMON_HOME 和 HADOOP_MAPRED_HOME，修改内容如下

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

拷贝 mysql-connector-java-5.0.7-bin.jar 到 \$SQOOP_HOME/lib 目录下

■ 验证 sqoop

以下命令用来验证 sqoop 版本

```
cd $SQOOP_HOME/bin
```

```
sqoop-version
```

预期结果如下：

```
15/07/01 02:09:27 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
```

```
Sqoop 1.4.6
```

```
git commit id c0c5a81723759fa575844a0a1eae8f510fa32c25
```

```
Compiled by root on Mon Apr 27 14:38:36 CST 2015
```

数据导入

本章介绍了如何从 MySQL 数据库中的数据导入到 Hadoop 的 HDFS。“导入工具”导入单个表从 RDBMS 到 HDFS。表中的每一行被视为 HDFS 的记录。所有记录都存储为文本文件的文本数据或二进制数据。

语法：

```
sqoop import (generic-args) (import-args)
```

import ##表示导入

```
--connect jdbc:mysql://ip:3306/sqoop ##告诉 jdbc，连接 mysql 的 url
```

```
--username root ##连接 mysql 的用户名
```

```
--password admin ##连接 mysql 的密码
```

```
--table aa ##从 mysql 导出的表名称
```

```
--fields-terminated-by '\t' ##指定输出文件中的行的字段分隔符
```

```
-m 1 ##复制过程使用 1 个 map 作业
```

命令的所有内容不能换行，只能一行才能执行。

任务执行结束后，观察 hdfs 的目录/user/{USER_NAME}，下面会有一个文件夹是 aa，里面有个文件是 part-m-00000。该文件的内容就是数据表 aa 的内容，字段之间是使用制表符分割的。

导入参数

参数	说明
- append	将数据追加到 hdfs 中已经存在的 dataset 中。使用该参数，sqoop 将把数据先导入到一个临时目录中，然后重新给文件命名到一个正式的目录中，以避免和该目录中已存在的文件重名。
- as-avrodatafile	将数据导入到一个 Avro 数据文件中
- as-sequencefile	将数据导入到一个 sequence 文件中
- as-textfile	将数据导入到一个普通文本文件中，生成该文本文件后，可以在 hive 中通过 sql 语句查询出结果。
- boundary-query <statement>	边界查询，也就是在导入前先通过 SQL 查询得到一个结果集，然后导入的数据就是该结果集内的数据，格式如： - boundary-query 'select id,creationdate from person where id = 3'，表示导入的数据为 id=3 的记录，或者 select min(<split-by>), max(<split-by>) from <table name>，注意查询的字段中不能有数据类型为字符串的字段，否则会报错： java.sql.SQLException: Invalid value for getLong()

	目前问题原因还未知
- columns<col,col,col...>	指定要导入的字段值，格式如： - columns id,username
- direct	直接导入模式，使用的是关系数据库自带的导入导出工具。官网上是说这样导入会更快
- direct-split-size	在使用上面 direct 直接导入的基础上，对导入的流按字节数分块，特别是使用直连模式从 PostgreSQL 导入数据的时候，可以将一个到达设定大小的文件分为几个独立的文件。
- inline-lob-limit	设定大对象数据类型的最大值
-m, - num-mappers	启动 N 个 map 来并行导入数据，默认是 4 个，最好不要将数字设置为高于集群的节点数
- query, -e<statement>	从查询结果中导入数据，该参数使用时必须指定 - target-dir、- hive-table，在查询语句中一定要有 where 条件且在 where 条件中需要包含\$CONDITIONS，示例： - query 'select * from person where \$CONDITIONS' - target-dir /user/hive/warehouse/person - hive-table person

- split-by<column-name>	表的列名，用来切分工作单元，一般后面跟主键 ID
- table <table-name>	关系数据库表名，数据从该表中获取
- target-dir <dir>	指定 hdfs 路径
- warehouse-dir <dir>	与 - target-dir 不能同时使用，指定数据导入的存放目录，适用于 hdfs 导入，不适合导入 hive 目录
- where	从关系数据库导入数据时的查询条件，示例：- where 'id = 2'
-z, - compress	压缩参数，默认情况下数据是未被压缩的，通过该参数可以使用 gzip 压缩算法对数据进行压缩，适用于 SequenceFile, text 文本文件，和 Avro 文件
- compression-codec	Hadoop 压缩编码，默认是 gzip
- null-string <null-string>	可选参数，如果没有指定，则字符串 null 将被使用
- null-non-string<null-string>	可选参数，如果没有指定，则字符串 null 将被使用

■ 导入到目标目录

在导入表数据到 HDFS 使用 Sqoop 导入工具，我们可以指定目标目录。

以下是指定目标目录选项的 Sqoop 导入命令的语法。

--target-dir <new or exist directory in HDFS>

```
sqoop import --connect jdbc:mysql://192.168.1.11/database --username root --password niit--table
sys_address_book --target-dir /home/hadoop/
```

■ 表数据导入子集

我们可以导入表的使用 Sqoop 导入工具, "where"子句的一个子集。它执行在各自的数据库服务器相应的 SQL 查询, 并将结果存储在 HDFS 的目标目录。

where 子句的语法如下。

```
--where <condition>
```

```
sqoop import --connect jdbc:mysql://192.168.1.11/database --username root --password niit--table  
sys_address_book --where 'id>100' --target-dir /home/hadoop/
```

■ 增量导入

增量导入是仅导入新添加的表中的行的技术。它需要添加 'incremental', 'check-column', 和 'last-value' 选项来执行增量导入。

下面的语法用于 Sqoop 导入命令增量选项。

```
--incremental <mode>
```

```
--check-column <column name>
```

```
--last value <last check column value>
```

```
sqoop import
```

```
--connect jdbc:mysql://192.168.1.11/database
```

```
--username root
```

```
--password niit
```

```
--table sys_address_book
```

```
--target-dir /home/hadoop/
```

```
--incremental append
```

```
--check-column id
```

```
--last value 1205
```

■ 导入所有表

如何导入从 RDBMS 数据库服务器到 HDFS 所有表。每个表的数据存储在一个单独的目录, 目录名与表名相同

语法:

```
sqoop import-all-tables --connect jdbc:mysql://localhost/userdb --username root
```

注: 如果使用的是 import-all-tables, 它是强制性的, 在该数据库中的每个表必须有一个主键字段。

■ 数据库连接参数

参数	说明
- connect <jdbc-uri>	Jdbc 连接 url, 例如: - connect jdbc:mysql://localhost:3306/hive
- connection-manager <class-name>	指定要使用的连接管理类
- driver <class-name>	数据库驱动类
- hadoop-home <dir>	Hadoop 根目录
- help	打印帮助信息
-P	从控制端读取密码
- password <password>	Jdbc url 中的数据库连接密码
- username <username>	Jdbc url 中的数据库连接用户名
- verbose	在控制台打印出详细信息
- connection-param-file <filename>	一个记录着数据库连接参数的文件
--default-character-set	设置字符集, 例如--default-character-set=utf-8

✧ 例如全表查询

```
sqoop import --connect jdbc:mysql://192.168.1.11/database --username root --password niit--table sys_address_book
```

✧ 例如自由式查询

```
sqoop import --connect jdbc:mysql://192.168.1.111/log --username root --password niit --query 'select * from log where id>24005 and $CONDITIONS' --split-by 'id' --target-dir /home/hadoop/target/ -m 1
```

Sqoop 还支持将任意的查询结果集导入，不使用--table、--columns 和--where，使用 SQL 语句--query 参数执行自由查询导入，但是必须指定--target-dir 目录，必须指定--split-by 分隔列，同时必须使用 where 且在其后加个\$CONDITIONS，使 Sqoop 进程替代为一个唯一的条件表达式达到条件查询效果

Sqoop 从大部分的数据源并行的导入数据，我们可以使用-m 参数控制 Map tasks 的数目，默认是 4 个，此处我们改成了 1 个 Map task。Map task,根据整个范围的均衡大小进行操作。例如，你有一张表，关键字 id 范围是 0-1000，默认 Map tasks 是 4 个，Sqoop 将会执行 4 个进程，每个进程以如下格式执行 SELECT * FROM sometable WHERE id >= lo AND id < hi 其中(lo, hi) set to (0, 250), (250, 500), (500, 750), and (750, 1001) 在不同的任务中。

注意： 如果你的表中关键字不是根据其范围均匀的分布，就可能导致不平衡的任务。这个时候你需要明确的选择一个不同的列使用--split-by 指定分隔参数。目前，Sqoop，还不支持对各个列索引进行分隔，如果一个表没有索引列或者含有多个关键字列，你必须手动的指定一个分隔列。

导入到 hive 中

如果数据导入 hive 中，可用以下参数

参数	说明
- hive-delims-replacement <arg>	用自定义的字符串替换掉数据中的\n, \r, and \01 等字符
- hive-drop-import-delims	在导入数据到 hive 中时，去掉数据中\n, \r 和\01 这样的字符
- map-column-hive <arg>	生成 hive 表时，可以更改生成字段的数据类型，格式如： - map-column-hiveTBL_ID=String, LAST_ACCESS_TIME=string

- hive-partition-key	创建分区，后面直接跟分区名即可，创建完毕后，通过 describe 表名可以看到分区名，默认为 string 型
- hive-partition-value<v>	该值是在导入数据到 hive 中时，与 - hive-partition-key 设定的 key 对应的 value 值。
- hive-home <dir>	Hive 的安装目录，可以通过该参数覆盖掉默认的 hive 目录
- hive-import	将数据从关系数据库中导入到 hive 表中
- hive-overwrite	覆盖掉在 hive 表中已经存在的数据
- create-hive-table	默认是 false, 如果目标表已经存在了，那么创建任务会失败
- hive-table	后面接要创建的 hive 表
- table	指定关系数据库表名

例如：

```
sqoop import --connect jdbc:mysql://192.168.1.11/database --username root --password niit--table
sys_address_book --hive-import
```

数据导出

将数据从 HDFS 导出到 RDBMS 数据库。目标表必须存在于目标数据库中。这是作为输入到 Sqoop 的文件包含记录，这被称为在表中的行。那些被读取并解析成一组记录和分隔与用户指定的分隔符。默认的操作是从输入文件到数据库表，使用 INSERT 语句插入所有记录。在更新模式，Sqoop 生成替换现有记录到数据库的 UPDATE 语句。

导出数据格式如下：


```
sqoop export --connect jdbc:mysql://192.168.1.111/database --username root --password
niit --table sa_menu_bak --export-dir /home/hadoop/target/
```

参数	说明
- enclosed-by <char>	给字段值前后加上指定的字符，比如双引号，示例： - enclosed-by “\” ‘，显示例子：” 3” ,” jimsss” ,” dd@dd.com”
- escaped-by <char>	给双引号作转义处理，如字段值为” 测试”，经过 - escaped-by \\处理后，在 hdfs 中的显示值为：\” 测试\”，对单引号无效
- fields-terminated-by <char>	设定每个字段是以什么符号作为结束的，默认是逗号，也可以改为其它符号，如句号.，示例如： - fields-terminated-by.
- lines-terminated-by <char>	设定每条记录行之间的分隔符，默认是换行，但也可以设定自己所需要的字符串，示例如： - lines-terminated-by ‘#’ 以#号分隔
- mysql-delimiters	Mysql 默认的分隔符设置，字段之间以, 隔开，行之间以换行\n 隔开，默认转义符号是\，字段值以单引号’ 包含起来。
- optionally-enclosed-by <char>	enclosed-by 是强制给每个字段值前后都加上指定的符号，而 - optionally-enclosed-by 只是给带有双引号或单引号的字段值加上指定的符号，故叫可选的。示例如： - optionally-enclosed-by ‘\$’ 显示结果：

	\$” hehe” , 测试\$
--	------------------

Sqoop 常用命令

1) 列出 mysql 数据库中的所有数据库

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

2) 连接 mysql 并列出 test 数据库中的表

```
sqoop list-tables -connect jdbc:mysql://localhost:3306/test -username root -password 123456
```

命令中的 test 为 mysql 数据库中的 test 数据库名称 username password 分别为 mysql 数据库的用户密码

3) 将关系型数据的表结构复制到 hive 中, 只是复制表的结构, 表中的内容没有复制过去。

```
sqoop create-hive-table -connect jdbc:mysql://localhost:3306/test  
-table sqoop_test -username root -password 123456 -hive-table test
```

其中 -table sqoop_test 为 mysql 中的数据库 test 中的表 -hive-table test 为 hive 中新建的表名称

4) 从关系数据库导入文件到 hive 中

```
sqoop import -connect jdbc:mysql://localhost:3306/zxtest -username  
root -password 123456 -table sqoop_test -hive-import -hive-table s_test -m 1
```

5) 将 hive 中的表数据导入到 mysql 中, 在进行导入之前, mysql 中的表 hive_test 必须已经提前创建好了。

```
sqoop export -connect jdbc:mysql://localhost:3306/zxtest -username  
root -password root -table hive_test -export-dir  
/user/hive/warehouse/new_test_partition/dt=2012-03-05
```

6) 从数据库导出表的数据到 HDFS 上文件

```
./sqoop import -connect  
jdbc:mysql://10.28.168.109:3306/compression -username=hadoop  
-password=123456 -table HADOOP_USER_INFO -m 1 -target-dir /user/test
```

7) 从数据库增量导入表数据到 hdfs 中

```
./sqoop import -connect jdbc:mysql://10.28.168.109:3306/compression
```

```
-username=hadoop -password=123456 -table HADOOP_USER_INFO -m 1  
-target-dir /user/test -check-column id -incremental append  
-last-value 3
```

Sqoop 高级

本章节将介绍 Sqoop 的高级部分，主要介绍 sqoop 的工作流程和优化点

■ Sqoop 原理（以 import 为例）

Sqoop 在 import 时，需要制定 split-by 参数。Sqoop 根据不同的 split-by 参数值来进行切分，然后将切分出来的区域分配到不同 map 中。每个 map 中再处理数据库中获取的一行一行的值，写入到 HDFS 中。

同时 split-by 根据不同的参数类型有不同的切分方法，如比较简单的 int 型，Sqoop 会取最大和最小 split-by 字段值，然后根据传入的 num-mappers 来确定划分几个区域。比如 select max(split_by),min(split-by) from 得到的 max(split-by)和 min(split-by)分别为 1000 和 1，而 num-mappers 为 2 的话，则会分成两个区域(1, 500)和(501-100)，同时也会分成 2 个 sql 给 2 个 map 去进行导入操作，分别为 select XXX from table where split-by>=1 and split-by<500 和 select XXX from table where split-by>=501 and split-by<=1000。最后每个 map 各自获取各自 SQL 中的数据进行导入工作。

■ mapreduce job 所需要的各种参数在 Sqoop 中的实现

1) InputFormatClass

com.cloudera.sqoop.mapreduce.db.DataDrivenDBInputFormat

2) OutputFormatClass

1)TextFile

com.cloudera.sqoop.mapreduce.RawKeyTextOutputFormat

2)SequenceFile

org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat

3)AvroDataFile

com.cloudera.sqoop.mapreduce.AvroOutputFormat

3)Mapper

1)TextFile

com.cloudera.sqoop.mapreduce.TextImportMapper

2) SequenceFile

```
com.cloudera.sqoop.mapreduce.SequenceFileImportMapper
```

3) AvroDataFile

```
com.cloudera.sqoop.mapreduce.AvroImportMapper
```

4) taskNumbers

```
1) mapred.map.tasks (对应 num-mappers 参数)
```

```
2) job.setNumReduceTasks(0);
```

■ 大概流程

1. 读取要导入数据的表结构，生成运行类，默认是 QueryResult，打成 jar 包，然后提交给 Hadoop

2. 设置好 job，主要也就是设置好以上第六章中的各个参数

3. 这里就由 Hadoop 来执行 MapReduce 来执行 Import 命令了，

1) 首先要对数据进行切分，也就是 DataSplit

```
DataDrivenDBInputFormat.getSplits(JobContext job)
```

2) 切分好范围后，写入范围，以便读取

```
DataDrivenDBInputFormat.write(DataOutput output) 这里是 lowerBoundQuery and  
upperBoundQuery
```

3) 读取以上 2) 写入的范围

```
DataDrivenDBInputFormat.readFields(DataInput input)
```

4) 然后创建 RecordReader 从数据库中读取数据

```
DataDrivenDBInputFormat.createRecordReader(InputSplit split, TaskAttemptContext context)
```

5) 创建 Map

```
TextImportMapper.setup(Context context)
```

6) RecordReader 一行一行从关系型数据库中读取数据，设置好 Map 的 Key 和 Value，交给 Map

```
DBRecordReader.nextKeyValue()
```

7) 运行 map

```
TextImportMapper.map(LongWritable key, SqoopRecord val, Context context)
```

最后生成的 Key 是行数据，由 QueryResult 生成，Value 是 NullWritable.get()

■ 总结

通过这些, 了解了 MapReduce 运行流程. 但对于 Sqoop 这种切分方式感觉还是有很大的问题. 比如这里根据 ID 范围来切分, 如此切分出来的数据会很不平均, 比如 $\min(\text{split-id})=1, \max(\text{split-id})=3000$, 交给三个 map 来处理. 那么范围是 (1-1000), (1001-2000), (2001-3000). 而假如 1001-2000 是没有数据, 已经被删除了. 那么这个 map 就什么都不能做. 而其他 map 却累的半死. 如此就会拖累 job 的运行结果. 这里说的范围很小, 比如有几十亿条数据交给几百个 map 去做. map 一多, 如果任务不均衡就会影响进度. 看有没有更好的切分方式? 比如取样? 如此看来, 写好 map reduce 也不简单!、



活动 1.1: 安装 **Sqoop** 练习 **sqoop** 导入导出

练习问题

1、关于 sqoop 描述错误的是 ()

- A、Sqoop 是 Hadoop 和关系数据库服务器之间传送数据的一种工具
- B、Sqoop 用来从关系数据库如：MySQL，Oracle 导入数据到 Hadoop 的 HDFS，并从 Hadoop 的文件系统导出数据到关系数据库
- C、Sqoop 将 SQL 语句解析为 MapReduce 后提交到 hadoop 集群中
- D、Sqoop 会自动选举 Leader 作为数据源导入者

2、关于 Sqoop 命令描述不正确的是 ()

- A、sqoop import --connect 是将关系型数据库导入到 hdfs 上
- B、sqoop import-all-tables --connect 导出关系型数据库上所有的表
- C、sqoop import --connect 。。。--query 'SQL ' 将符合条件的数据导入 hdfs
- D、以上都不对

3、关于 sqoop 导入导出命令使用错误的是

- A、sqoop export --connect jdbc:mysql://192.168.1.111/database --username root --password niit --table sa_menu_bak --export-dir /home/hadoop/target/
- B、sqoop create-hive-table - connect jdbc:mysql://localhost:3306/test - table sqoop_test -username root -password 123456 -hive-table test
- C、sqoop export - connect jdbc:mysql://localhost:3306/zxtest - username root - password root - table hive_test - export-dir /user/hive/warehouse/new_test_partition/dt=2012-03-05
- D、./sqoop import - connect jdbc:mysql://10.28.168.109:3306/compression - username=hadoop - password=123456 - table HADOOP_USER_INFO -m 1 - disk /user/test

小结

在本章中，您已学习了流式计算的相关知识：

- ✧ Sqoop 概念
- ✧ Sqoop 安装
- ✧ Sqoop 导入
- ✧ Sqoop 导出
- ✧ Sqoop 工作原理
- ✧ Sqoop 优化

Sqoop 在大数据分析中非常重要，通过用来将 hive 中产生的已经汇总的数据导入到 MySQL 中。Sqoop 的命令应该经常联系，模拟不通情况来导入导出。

练习答案

第 1 章

1. D
2. D
3. D

参考阅读



大数据工具简介

下表列出了《大数据生态圈工具》教程的参考资料。

编号	书名	网址
Sqoop		http://sqoop.apache.org/#

参考资料列表

免责声明：“参考阅读”部分列出的所有 URL 的准确性和适当性均在添加时经过检验。但不能保证其网站和内容长时间不变。

