

Sqoop 数据迁移工具

目录

1、概述	1
2、工作机制	2
3、Sqoop 安装	2
4、Sqoop 基本命令介绍	4
5、Sqoop 数据导入	4
5.1、导入 MySQL 数据到 HDFS	5
5.1.1、普通导入	5
5.1.2、指定分隔符和导入路径	7
5.1.3、导入 where 条件数据	8
5.1.4、导入 Query 结果数据	9
5.2、导入 MySQL 到 HIVE	10
5.2.1、普通导入	10
5.2.2、增量导入	12
5.3、导入 MySQL 数据到 HBase	13
6、Sqoop 数据导出	14
6.1、导出 HDFS 数据到 MySQL	14
6.2、导出 HIVE 数据到 MySQL	16
6.3、导出 HBase 数据到 MySQL	17
7、Sqoop 导入导出的原理剖析	17
7.1、Sqoop 导入原理	17
7.2、Sqoop 导出原理	18
8、资料	19

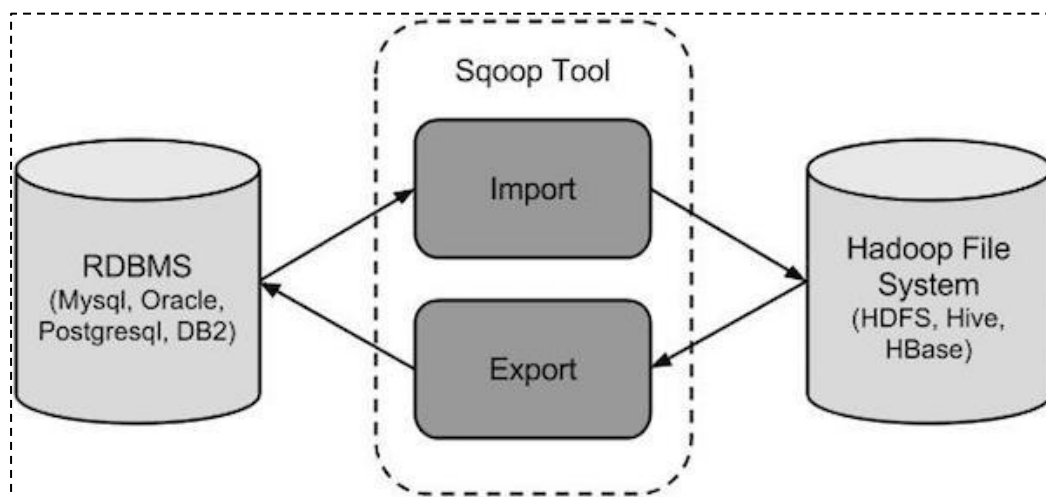
1、概述

sqoop 是 apache 旗下一款 “Hadoop 和关系数据库服务器之间传送数据” 的工具。

导入数据: MySQL, Oracle 导入数据到 Hadoop 的 HDFS、HIVE、HBASE 等数据存储系统

导出数据: 从 Hadoop 的文件系统中导出数据到关系数据库 mysql 等

Sqoop 的本质还是一个命令行工具，和 HDFS，Hive 相比，并没有什么高深的理论。



2、工作机制

将导入或导出命令翻译成 MapReduce 程序来实现
在翻译出的 MapReduce 中主要是对 InputFormat 和 OutputFormat 进行定制

3、Sqoop 安装

安装 Sqoop 的前提是已经具备 Java 和 Hadoop 的环境

安装包下载地址 <http://ftp.wayne.edu/apache/sqoop/1.4.6/>

安装包: sqoop-1.4.6.bin_hadoop-2.0.4-alpha.tar.gz

安装步骤

- 1、准备安装包 sqoop-1.4.6.bin_hadoop-2.0.4-alpha.tar.gz
- 2、解压安装包到安装目录

```
tar -zxvf sqoop-1.4.6.bin_hadoop-2.0.4-alpha.tar.gz -C apps/
cd apps
mv sqoop-1.4.6.bin_hadoop-2.0.4-alpha/ sqoop-1.4.6
```
- 3、进入到 conf 文件夹，找到 sqoop-env-template.sh，修改其名称为 sqoop-env.sh

```
cd conf
mv sqoop-env-template.sh sqoop-env.sh
```
- 4、修改 sqoop-env.sh

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=/home/hadoop/apps/hadoop-2.6.5

#Set path to where hadoop-*-core.jar is available
export HADOOP_MAPRED_HOME=/home/hadoop/apps/hadoop-2.6.5

#set the path to where bin/hbase is available
export HBASE_HOME=/home/hadoop/apps/hbase-1.2.6

#Set the path to where bin/hive is available
export HIVE_HOME=/home/hadoop/apps/apache-hive-1.2.1-bin

#Set the path for where zookeeper config dir is
export ZOOCFGDIR=/home/hadoop/apps/zookeeper-3.4.10/conf
```

```
export HADOOP_COMMON_HOME=/home/hadoop/apps/hadoop-2.6.5
export HADOOP_MAPRED_HOME=/home/hadoop/apps/hadoop-2.6.5
export HBASE_HOME=/home/hadoop/apps/hbase-1.2.6
export HIVE_HOME=/home/hadoop/apps/apache-hive-1.2.1-bin
export ZOOCFGDIR=/home/hadoop/apps/zookeeper-3.4.10/conf
```

zookeeper 和 hbase 没有安装。那就不用管了。如果也安装的有，并且要使用，那么就给配置上

5、加入 mysql 驱动包到 sqoop1.4.6/lib 目录下

cp mysql-connector-java-5.1.40-bin.jar ~/apps/sqoop1.4.6/lib/

6、配置系统环境变量

vi ~/.bashrc

然后输入：

```
export SQOOP_HOME=/home/hadoop/apps/sqoop1.4.6
```

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

然后保存退出

```
source ~/.bashrc
```

7、验证安装是否成功

sqoop-version 或者 **sqoop version**

```
[hadoop@hadoop05 lib]$ sqoop-version
Warning: /home/hadoop/apps/sqoop-1.4.6/bin/../../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/hadoop/apps/sqoop-1.4.6/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/hadoop/apps/sqoop-1.4.6/bin/../../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
17/03/07 10:30:36 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
```

8、出现如图所示画面，证明安装成功，那么接下来就可以正常使用了。

4、Sqoop 基本命令介绍

首先，我们可以使用 `sqoop help` 来查看，sqoop 支持哪些命令

Available commands:

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
help	List available commands
import	Import a table from a database to HDFS
import-all-tables	Import tables from a database to HDFS
import-mainframe	Import datasets from a mainframe server to HDFS
job	Work with saved jobs
list-databases	List available databases on a server
list-tables	List available tables in a database
merge	Merge results of incremental imports
metastore	Run a standalone Sqoop metastore
version	Display version information

然后得到这些支持的命令之后，如果不知道使用方式，可以使用 `sqoop command` 的方式来查看某条具体命令的使用方式，比如：

sqoop help import

```
Common arguments:
--connect <jdbc-uri>          Specify JDBC connect
                              string
--connection-manager <class-name> Specify connection manager
                              class name
--connection-param-file <properties-file> Specify connection
                              parameters file
--driver <class-name>        Manually specify JDBC
                              driver class to use
--hadoop-home <hdir>         Override
                              $HADOOP_MAPRED_HOME_ARG
--hadoop-mapred-home <dir>   Override
                              $HADOOP_MAPRED_HOME_ARG
--help                        Print usage instructions
-P                             Read password from console
--password <password>       Set authentication
                              password
```

这里只给出了一部分参数的截图，下面的例子演示，我会教大家来使用常用的一些命令参数

5、Sqoop 数据导入

“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（或者 Avro、sequence 文件等二进制数据）

下面的语法用于将数据导入 HDFS

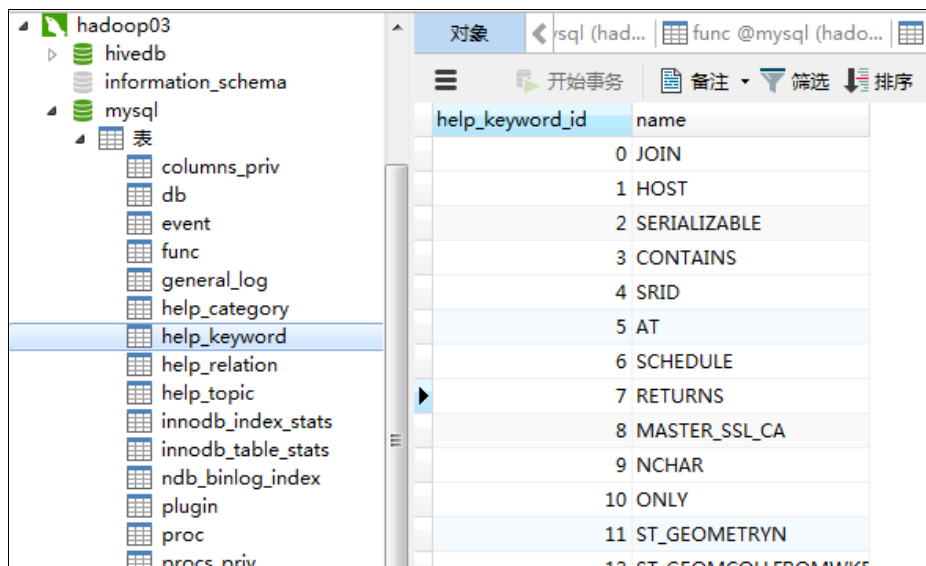
`sqoop import (generic-args) (import-args)`

常用参数:

<code>--connect <jdbc-uri></code>	JDBC 连接地址
<code>--connection-manager <class-name></code>	连接管理者
<code>--driver <class-name></code>	驱动类
<code>--hadoop-mapred-home <dir></code>	\$HADOOP_MAPRED_HOME
<code>--help</code>	help 信息
<code>-P</code>	从命令行输入密码
<code>--password <password></code>	密码
<code>--username <username></code>	账号
<code>--verbose</code>	打印流程信息
<code>--connection-param-file <filename></code>	可选参数

示例:

先看 MySQL 表数据: 这是 mysql 中 help_keyword 表的数据



The screenshot shows a MySQL database interface with a tree view on the left and a table view on the right. The tree view shows the hierarchy: hadoop03 > hivedb > information_schema > mysql > 表 (Tables). The 'help_keyword' table is selected. The table view shows the following data:

help_keyword_id	name
0	JOIN
1	HOST
2	SERIALIZABLE
3	CONTAINS
4	SRID
5	AT
6	SCHEDULE
7	RETURNS
8	MASTER_SSL_CA
9	NCHAR
10	ONLY
11	ST_GEOMETRYN
12	ST_GEOMCOLLFROMWKE

5.1、导入 MySQL 数据到 HDFS

5.1.1、普通导入

```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
-m 1
```

如果我们没有给该命令指定导出的文件的存储路径, 那么默认会保存在 HDFS 上的

`/user/hadoop/help_keyword` 目中

其中，第一个 user 是固定的，第二个 hadoop，表示链接的用户名，第三个表示表名

查看结果：

执行过程结果：

```

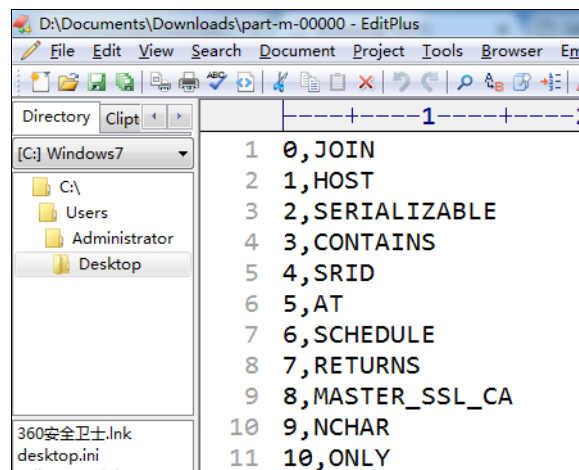
Map-Reduce Framework
  Map input records=608
  Map output records=608
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=106
  CPU time spent (ms)=940
  Physical memory (bytes) snapshot=100519936
  Virtual memory (bytes) snapshot=846106624
  Total committed heap usage (bytes)=18427904
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=8136
16/11/20 22:41:39 INFO mapreduce.ImportJobBase: Transferred 7.9453 KB in 27.4485 seconds (296.4097 bytes/sec)
16/11/20 22:41:39 INFO mapreduce.ImportJobBase: Retrieved 608 records.
[root@hadoop01 sqoop1.4.6]#

```

HDFS 上文件结果：

Browse Directory						
/user/root/help_keyword						Go!
Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	2	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	7.95 KB	2	128 MB	part-m-00000

可以下载下来使用本地文本编辑器查看比对结果（文件内容结果）



也可以使用 hadoop 命令查看：

`hadoop fs -cat /user/hadoop/help_keyword/part-m-00000`

从以上结果可以得出一个结论：如果没有指定路径，则会按默认规则生成路径，如果没有指定分隔符，默认按照逗号分隔

5.1.2、指定分隔符和导入路径

先看指定导入路径的语法规则：

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

再看指定导入的文本文件的列分隔符：

```
--fields-terminated-by '\t'
```

看具体实例：

```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
--target-dir /user/hadoop/my_help_keyword \
--fields-terminated-by '\t' \
-m 1
```

看结果：

```
Map-Reduce Framework
  Map input records=608
  Map output records=608
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=110
  CPU time spent (ms)=940
  Physical memory (bytes) snapshot=103837696
  Virtual memory (bytes) snapshot=846106624
  Total committed heap usage (bytes)=18165760
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=8136
16/11/20 22:55:58 INFO mapreduce.ImportJobBase: Transferred 7.9453 KB in 23.1617 seconds (351.2698 bytes/sec)
16/11/20 22:55:58 INFO mapreduce.ImportJobBase: Retrieved 608 records.
[root@hadoop01 sqoop1.4.6]#
```

Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	2	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	7.95 KB	2	128 MB	part-m-00000

Hadoop, 2016.

```
hadoop fs -cat /user/hadoop/my_help_keyword/part-m-00000
```

1	0	JOIN
2	1	HOST
3	2	SERIALIZABLE
4	3	CONTAINS
5	4	SRID
6	5	AT
7	6	SCHEDULE
8	7	RETURNS
9	8	MASTER_SSL_CA
10	9	NCHAR
11	10	ONLY

5.1.3、导入 where 条件数据

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>
```

下面我们看具体实例:

```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--where "name='STRING' " \
--table help_keyword \
--target-dir /sqoop/hadoop/myoutport1 \
-m 1
```

看执行结果:

```
16/11/20 23:18:45 INFO mapreduce.ImportJobBase: Retrieved 1 records.
[root@hadoop01 sqoop1.4.6]#
Map-Reduce Framework
  Map input records=1
  Map output records=1
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=104
  CPU time spent (ms)=910
  Physical memory (bytes) snapshot=105586688
  Virtual memory (bytes) snapshot=846106624
  Total committed heap usage (bytes)=18579456
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=11
16/11/20 23:18:45 INFO mapreduce.ImportJobBase: Transferred 11 bytes in 21.3386 seconds (0.5155 bytes/sec)
16/11/20 23:18:45 INFO mapreduce.ImportJobBase: Retrieved 1 records.
```

看导出的文本结果:

```
hadoop fs -cat /sqoop/hadoop/myoutport1/part-m-00000
```



```

hadoop02 | hadoop03 | hadoop04 | hadoop05 x
[hadoop@hadoop05 ~]$ hadoop fs -cat /sqoop/hadoop/myoutport1/part-m-00000
102,STRING
[hadoop@hadoop05 ~]$

```

5.1.4、导入 Query 结果数据

```

sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--target-dir /user/hadoop/myimport2 \
--query 'select help_keyword_id,name from help_keyword WHERE name = "STRING" and $CONDITIONS' \
--split-by help_keyword_id \
--fields-terminated-by '\t' \
-m 1

```

注意：外层使用单引号，SQL 语句当中的条件使用双引号，否则会报错
或者这么写：

```

sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--target-dir /user/hadoop/myimport22 \
--query "select help_keyword_id,name from help_keyword WHERE name = 'STRING' and \
$CONDITIONS" \
--split-by help_keyword_id \
--fields-terminated-by '\t' \
-m 1

```

查看执行结果：

```

Map-Reduce Framework
  Map input records=10
  Map output records=10
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=121
  CPU time spent (ms)=930
  Physical memory (bytes) snapshot=103141376
  Virtual memory (bytes) snapshot=846106624
  Total committed heap usage (bytes)=18636800
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=97
16/11/20 23:25:21 INFO mapreduce.ImportJobBase: Transferred 97 bytes in 22.8493 seconds (4.2452 bytes/sec)
16/11/20 23:25:21 INFO mapreduce.ImportJobBase: Retrieved 10 records.
[root@hadoop01 sqoop1.4.6]#

```

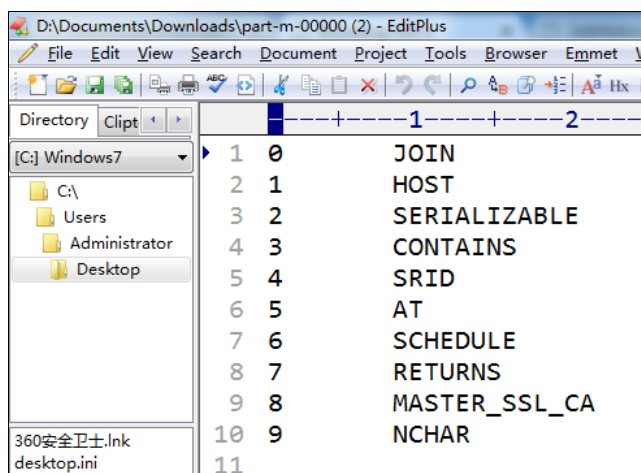
Browse Directory

/user/hadoop/myimport2

Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	2	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	97 B	2	128 MB	part-m-00000

Hadoop, 2016.



5.2、导入 MySQL 到 HIVE

Sqoop 导入关系型数据到 hive 的过程是先导入到 hdfs，然后再 load 进入 hive

5.2.1、普通导入

示例：

```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
--hive-import \
-m 1
```

注意：导入数据到 hive 表，默认表在 default 库下，表名一样，采用'\u0001'分隔查看执行结果：

```

16/11/20 23:02:51 INFO mapreduce.ImportJobBase: Transferred 7.9453 KB in 22.4503 seconds (362.4008 bytes/sec)
16/11/20 23:02:51 INFO mapreduce.ImportJobBase: Retrieved 608 records.
16/11/20 23:02:51 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `help_keyword` AS t LIMIT 1
16/11/20 23:02:51 INFO hive.HiveImport: Loading uploaded data into Hive
16/11/20 23:02:54 INFO hive.HiveImport: Logging initialized using configuration in jar:file:/root/apps/apache-hive-1.2.1
bin/lib/hive-common-1.2.1.jar!/hive-log4j.properties
16/11/20 23:03:04 INFO hive.HiveImport: OK
16/11/20 23:03:04 INFO hive.HiveImport: Time taken: 1.03 seconds
16/11/20 23:03:04 INFO hive.HiveImport: Loading data to table default.help_keyword
16/11/20 23:03:05 INFO hive.HiveImport: Table default.help_keyword stats: [numFiles=1, totalSize=8136]
16/11/20 23:03:05 INFO hive.HiveImport: OK
16/11/20 23:03:05 INFO hive.HiveImport: Time taken: 0.646 seconds
16/11/20 23:03:05 INFO hive.HiveImport: Hive import complete.
16/11/20 23:03:05 INFO hive.HiveImport: Export directory is contains the _SUCCESS file only, removing the directory.
[root@hadoop01 sqoop1.4.6]#

```

查看结果，确实生成结果在 hive 的 default 库中，查看表数据：

hive> select * from help_keyword limit 3;

```

hive> use default;
OK
Time taken: 0.07 seconds
hive> show tables;
OK
help_keyword
Time taken: 0.059 seconds, Fetched: 1 row(s)
hive> select * from help_keyword limit 3;
OK
0      JOIN
1      HOST
2      SERIALIZABLE
Time taken: 0.135 seconds, Fetched: 3 row(s)
hive>

```

接下来，再看一例：

```

sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
--fields-terminated-by "\t" \
--lines-terminated-by "\n" \
--hive-import \
--hive-overwrite \
--create-hive-table \
--hive-table mydb_test.new_help_keyword \
--delete-target-dir

```

注意：表会自动创建，但是库不会。所以在执行该语句之前，一定要确保 hive 的数据库 mydb_test 是存在的，否则程序会报错

查看执行结果：

Browse Directory						
/user/hive/warehouse/mydb_test.db/new_help_keyword						Go!
Permission	Owner	Group	Size	Replication	Block Size	Name
-rwxr-xr-x	hadoop	supergroup	1.88 KB	2	128 MB	part-m-00000
-rwxr-xr-x	hadoop	supergroup	2 KB	2	128 MB	part-m-00001
-rwxr-xr-x	hadoop	supergroup	2.06 KB	2	128 MB	part-m-00002
-rwxr-xr-x	hadoop	supergroup	2.01 KB	2	128 MB	part-m-00003

Hadoop, 2016.

```
hive> use mydb_test;
OK
Time taken: 0.083 seconds
hive> show tables;
OK
new_help_keyword
Time taken: 0.094 seconds, Fetched: 1 row(s)
hive> desc new_help_keyword;
OK
help_keyword_id      bigint
name                  string
Time taken: 0.326 seconds, Fetched: 2 row(s)
hive> select * from new_help_keyword limit 5;
OK
0      JOIN
1      HOST
2      SERIALIZABLE
3      CONTAINS
4      SRID
Time taken: 0.23 seconds, Fetched: 5 row(s)
hive>
```

5.2.2、增量导入

增量导入是仅导入表中新添加的行的技术。

它需要添加 '**incremental**', '**check-column**', 和 '**last-value**' 选项来执行增量导入。

下面的语法结构用于 Sqoop 导入命令增量选项。

```
--incremental <mode>
--check-column <column name>
--last value <last check column value>
```

下面的命令执行增量导入：

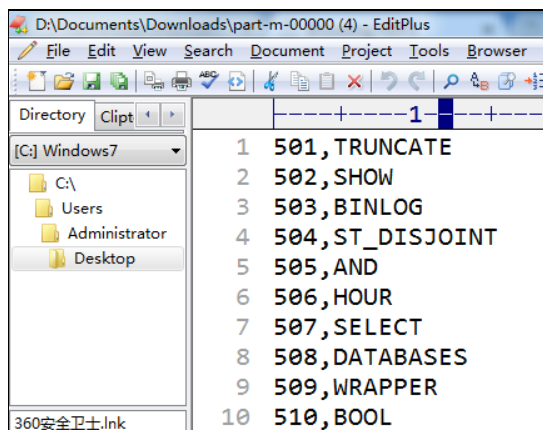
```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
--target-dir /user/hadoop/myimport3 \
--incremental append \
--check-column help_keyword_id \
--last-value 500 \
-m 1
```

此处结果不包含 help_keyword_id = 500 数据，数据是大于 500 的

查看执行结果：

```
16/11/20 23:50:32 INFO mapreduce.ImportJobBase: Transferred 1.4062 KB in 23.6487 seconds (60.8914 bytes/sec)
16/11/20 23:50:32 INFO mapreduce.ImportJobBase: Retrieved 107 records.
16/11/20 23:50:32 INFO util.AppendUtils: Creating missing output directory - myimport3
16/11/20 23:50:32 INFO tool.ImportTool: Incremental import complete! To run another incremental import of all data follow
ing this import, supply the following arguments:
16/11/20 23:50:32 INFO tool.ImportTool: --incremental append
16/11/20 23:50:32 INFO tool.ImportTool: --check-column help_keyword_id
16/11/20 23:50:32 INFO tool.ImportTool: --last-value 607
16/11/20 23:50:32 INFO tool.ImportTool: (consider saving this with 'sqoop job --create')
[root@hadoop01 ~]#
```

Browse Directory						
<input type="text" value="/user/hadoop/myimport3"/>						<input type="button" value="Go!"/>
Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	1.41 KB	2	128 MB	part-m-00000
Hadoop, 2016.						



5.3、导入 MySQL 数据到 HBase

首先看示例：

```
sqoop import \
--connect jdbc:mysql://hadoop02:3306/mysql \
--username root \
--password root \
--table help_keyword \
--hbase-table new_help_keyword \
--column-family person \
--hbase-row-key help_keyword_id
```

字段解释：

--connect jdbc:mysql://hadoop04:3306/mysql	表示远程或者本地 Mysql 服务的 URI
--hbase-create-table	表示在 HBase 中建立表。
--hbase-table new_help_keyword	表示在 HBase 中建立表 new_help_keyword。
--hbase-row-key help_keyword_id	表示 hbase 表的 rowkey 是 mysql 表的 help_keyword_id 字段。
--column-family person	表示在表 new_help_keyword 中建立列族 person。
--username 'root'	表示使用用户 root 连接 mysql。
--password 'root'	连接 mysql 的用户密码
--table help_keyword	表示导出 mysql 数据库的 help_keyword 表。

看结果：

```

85      column=person:name, timestamp=1499794316831, value=LONG
86      column=person:name, timestamp=1499794316831, value=OPTION
87      column=person:name, timestamp=1499794316831, value=REORGANIZE
88      column=person:name, timestamp=1499794316831, value=ELSE
89      column=person:name, timestamp=1499794316831, value=EXTERIORRING
90      column=person:name, timestamp=1499794316831, value=NCHAR
91      column=person:name, timestamp=1499794316831, value=GEOMFROMWKB
92      column=person:name, timestamp=1499794316831, value=STATS_AUTO_RECALC
93      column=person:name, timestamp=1499794316831, value=FROM
94      column=person:name, timestamp=1499794316831, value=MULTIPOLYGON
95      column=person:name, timestamp=1499794316831, value=LEFT
96      column=person:name, timestamp=1499794316831, value=ELSEIF
97      column=person:name, timestamp=1499794316831, value=ST_ISCLOSED
98      column=person:name, timestamp=1499794316831, value=COMPACT
99      column=person:name, timestamp=1499794316831, value=DEC
608 row(s) in 2.4340 seconds
hbase(main):008:0>

```

6、Sqoop 数据导出

6.1、导出 HDFS 数据到 MySQL

注意：导出前，目标表必须存在于目标数据库中。

HDFS: hadoop distributed file system

RDBMS: Relation DataBase Manager System

OOP: orentied object programming

AOP: orentied aspect programming

默认操作是从将文件中的数据使用 INSERT 语句插入到表中
更新模式下，是生成 UPDATE 语句更新表数据

Export 语法结构：

```
sqoop export (generic-args) (export-args)
```

Export 常用参数：

export 主要参数

--direct	快速导入
--export-dir <dir>	HDFS 到处数据的目录
-m,--num-mappers <n>	都少个 map 线程
--table <table-name>	导出哪个表
--call <stored-proc-name>	存储过程
--update-key <col-name>	通过哪个字段来判断更新
--update-mode <mode>	插入模式，默认是只更新，可以设置为 allowinsert.
--input-null-string <null-string>	字符类型 null 处理
--input-null-non-string <null-string>	非字符类型 null 处理
--staging-table <staging-table-name>	临时表
--clear-staging-table	清空临时表
--batch	批量模式

第一步，先看需要导出到 mysql 的数据：

```
[root@hadoop01 ~]# cat student.txt
95002,刘晨,女,19,IS
95017,王凤娟,女,18,IS
95018,王一,女,19,IS
95013,冯伟,男,21,CS
95014,王小丽,女,19,CS
95019,邢小丽,女,19,IS
95020,赵钱,男,21,IS
95003,王敏,女,22,MA
95004,张立,男,19,IS
95012,孙花,女,20,CS
95010,孔小涛,男,19,CS
95005,刘刚,男,18,MA
95006,孙庆,男,23,CS
95007,易思玲,女,19,MA
95008,李娜,女,18,CS
95021,周二,男,17,MA
95022,郑明,男,20,MA
95001,李勇,男,20,CS
95011,包小柏,男,18,MA
95009,梦圆圆,女,18,MA
95015,王君,男,18,MA
95016,钱国,男,21,MA
[root@hadoop01 ~]# hadoop fs -mkdir /sqoopdata
[root@hadoop01 ~]# hadoop fs -put student.txt /sqoopdata
[root@hadoop01 ~]#
```

然后第二步，需要首先在 mysql 数据库中新建表：

```
mysql> create database sqoopdb default character set utf8;
Query OK, 1 row affected (0.00 sec)

mysql> use sqoopdb;
Database changed
mysql> CREATE TABLE sqoopstudent (
  ->   id INT NOT NULL PRIMARY KEY,
  ->   name VARCHAR(20),
  ->   sex VARCHAR(20),
  ->   age INT,
  ->   department VARCHAR(20));
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
+-----+
| Tables_in_sqoopdb |
+-----+
| sqoopstudent       |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
create database sqoopdb default character set utf8 COLLATE utf8_general_ci;
use sqoopdb;
CREATE TABLE sqoopstudent (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(20),
    sex VARCHAR(20),
    age INT,
    department VARCHAR(20)
);
```

第三步，执行导出：

```
sqoop export \
--connect jdbc:mysql://hadoop02:3306/sqoopdb \
--username root \
--password root \
--table sqoopstudent \
--export-dir /sqoopdata \
--fields-terminated-by ','
```

查看结果:

```
Map-Reduce Framework
  Map input records=22
  Map output records=22
  Input split bytes=551
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=1120
  CPU time spent (ms)=3830
  Physical memory (bytes) snapshot=347963392
  Virtual memory (bytes) snapshot=3376005120
  Total committed heap usage (bytes)=74526720
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=0
16/11/21 00:22:23 INFO mapreduce.ExportJobBase: Transferred 1.9053 KB in 40.5145 seconds (48.1556 bytes/sec)
16/11/21 00:22:23 INFO mapreduce.ExportJobBase: Exported 22 records.
[root@hadoop01 sqoop1.4.6]#
```

```
hadoop02 | hadoop03 | hadoop04 x | hadoop05
[hadoop@hadoop04 ~]$ hadoop fs -cat /sqoopdata/student.txt
101,huangbo,男,22,CS
102,xuzheng,男,23,MA
103,liutao,女,24,CC
[hadoop@hadoop04 ~]$
```

```
mysql> select * from sqoopstudent;
+----+-----+-----+-----+-----+
| id | name  | sex  | age | department |
+----+-----+-----+-----+-----+
| 101 | huangbo | 男   | 22  | CS         |
| 102 | xuzheng | 男   | 23  | MA         |
| 103 | liutao  | 女   | 24  | CC         |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

注意: 如果导出数据到 mysql 出现乱码, 那么请参考这个解决方案

<https://my.oschina.net/u/559635/blog/498990>

6.2、导出 HIVE 数据到 MySQL

```
sqoop export \
--connect jdbc:mysql://hadoop02:3306/sqoopdb \
--username root \
--password root \
--table uv_info \
--export-dir /user/hive/warehouse/uv/dt=2011-08-03 \
--input-fields-terminated-by '\t'
```

其实跟直接导出 HDFS 数据到 MySQL 没什么两样

6.3、导出 HBase 数据到 MySQL

很遗憾，现在还没有直接的命令将 HBase 的数据导出到 MySQL
一般采用如下 3 种方法：

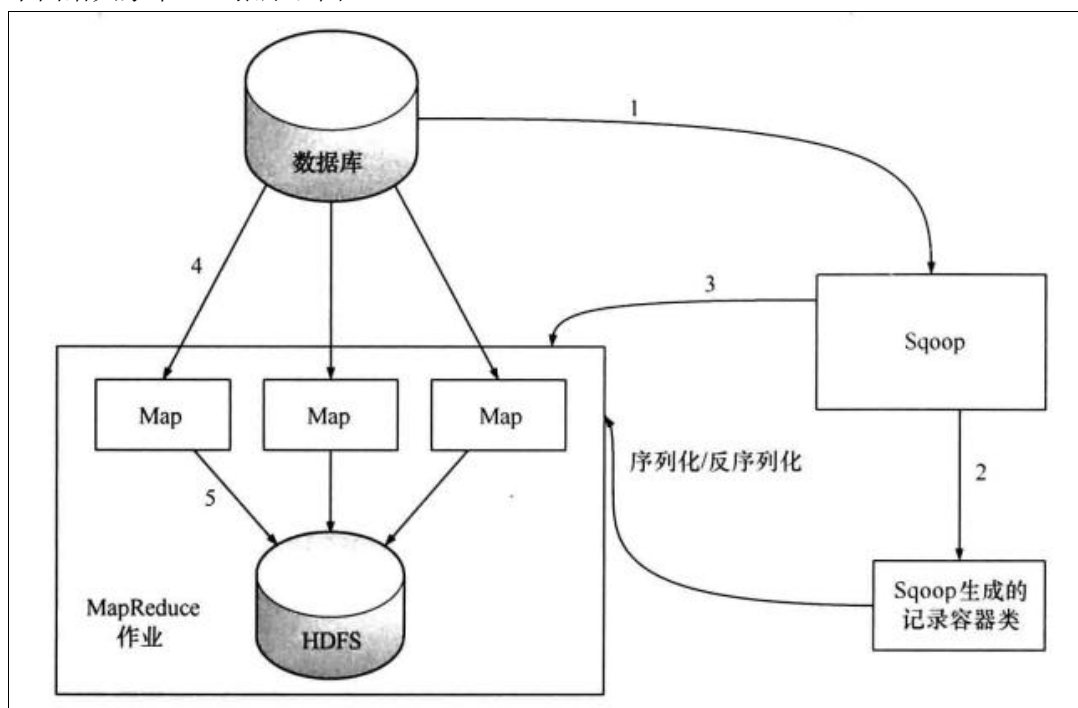
- 1、将 Hbase 数据，扁平化成 HDFS 文件，然后再由 sqoop 导入
- 2、将 Hbase 数据导入 Hive 表中，然后再导入 mysql
- 3、直接使用 Hbase 的 Java API 读取表数据，直接向 mysql 导入，不需要使用 sqoop

7、Sqoop 导入导出的原理剖析

7.1、Sqoop 导入原理

从上面的演示例子中，我们大致能得出一个结论，sqoop 工具是通过 MapReduce 进行导入作业的。总体来说，是把关系型数据库中的某张表的一行行记录都写入到 hdfs

下面给大家奉上一张原理图：



上面这张图大致解释了 sqoop 在进行数据导入工作的大致流程，下面我们用文字来详细描述一下：

1、第一步，Sqoop 会通过 JDBC 来获取所需要的数据库元数据，例如，导入表的列名，数据类型等。

2、第二步，这些数据库的数据类型(varchar, number 等)会被映射成 Java 的数据类型(String, int 等)，根据这些信息，Sqoop 会生成一个与表名同名的类来完成序列化工作，保存表中的

每一行记录。

3、第三步，Sqoop 启动 MapReducer 作业

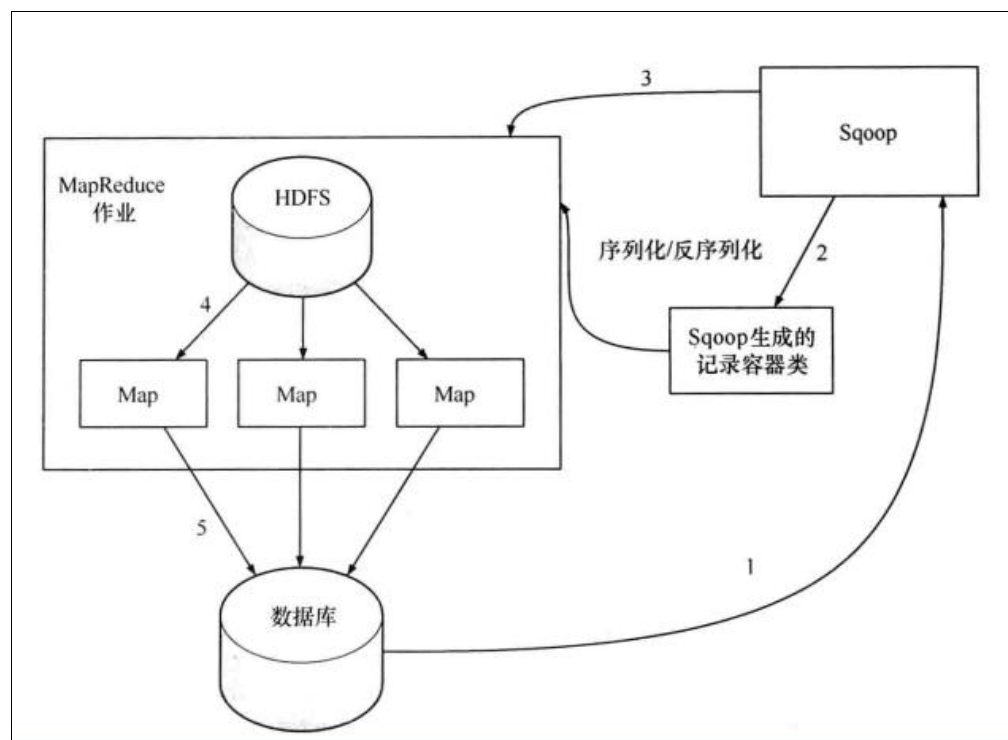
4、第四步，启动的作业在 input 的过程中，会通过 JDBC 读取数据表中的内容，这时，会使用 Sqoop 生成的类进行反序列化操作

5、第五步，最后将这些记录写到 HDFS 中，在写入到 HDFS 的过程中，同样会使用 Sqoop 生成的类进行反序列化

7.2、Sqoop 导出原理

Sqoop 进行数据导出，总体也是基于 mapreduce 任务。

下面先看图：



详细文字描述：

- 1、第一步，sqoop 依然会通过 JDBC 访问关系型数据库，得到需要导出数据的元数据信息
- 2、第二步，根据获取到的元数据的信息，sqoop 生成一个 Java 类，用来进行数据的传输载体。该类必须实现序列化和反序列化
- 3、第三步，启动 mapreduce 作业

- 4、第四步，sqoop 利用生成的这个 java 类，并行的从 hdfs 中读取数据
- 5、第五步，每个 map 作业都会根据读取到的导出表的元数据信息和读取到的数据，生成一批 insert 语句，然后多个 map 作业会并行的向数据库 mysql 中插入数据

所以，数据是从 hdfs 中并行的进行读取，也是并行的进入写入，那并行的读取是依赖 hdfs 的性能，而并行的写入到 mysql 中，那就要依赖于 mysql 的写入性能嘞。

8、资料

Sqoop 详细可以参照我的博客：

<http://blog.csdn.net/zhongqi2513/article/details/52777727>

<http://blog.csdn.net/zhongqi2513/article/details/53281255>

当然最重要的是官网文档：

<http://sqoop.apache.org/docs/>