目录

**[一. 课前提示 1](#_Toc6311)**

**[二. 正课内容 1](#_Toc16723)**

[1. Sqoop简介以及使用 1](#_Toc30101)

[2 Sqoop的安装 3](#_Toc18092)

[3.Sqoop命令执行 4](#_Toc24513)

[4.Sqoop导入实战 10](#_Toc11684)

[5.Sqoop导出 13](#_Toc32260)

[6.Sqoop优化 14](#_Toc3707)

# 一. 课前提示

### 1. 今日任务

1. 了解Sqoop的来历  
2. Sqoop的原理和底层实现  
3. Sqoop的配置安装  
4. Sqoop导入

### 2. 今日目标

2.1 了解和熟悉部分

- Sqoop的简介  
- Sqoop的底层原理  
- Sqoop的环境配置

2.2 **掌握部分**

- Sqoop和HDFS导入导出  
- Sqoop和Hive的导入导出  
- Sqoop和Hbase的导出

# 二. 正课内容

## 1. Sqoop简介以及使用

### 1.1 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。

### 1.2 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：

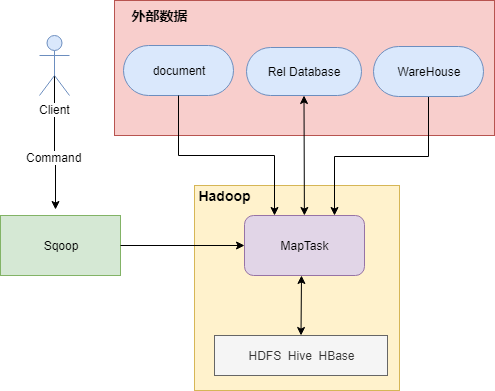
1. sqoop
2. datax
3. kettle
4. cannal

### 1.3 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.

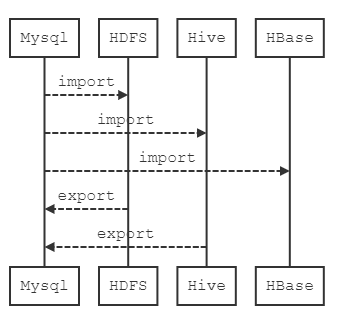
Sqoop结构图:



### 1.4 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。

## 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

### 2.1、解压配置环境变量

tar -zxvf /opt/soft/sqoop... -C /opt/app/sqoop...  
vi /etc/profile

### 2.2、新建配置文件

mv ./conf/sqoop-env-template.sh ./conf/sqoop-env.sh

### 2.3、修改配置文件

配置文件：

vi ./conf/sqoop-env.sh

export HADOOP\_COMMON\_HOME=/usr/local/hadoop-2.7.1/  
export HADOOP\_MAPRED\_HOME=/usr/local/hadoop-2.7.1/  
export HIVE\_HOME=/usr/local/hive-1.2.1/  
export ZOOCFGDIR=/usr/local/zookeeper-3.4.7/

### 2.4、拷贝mysql驱动

cp /home/mysql-connector-java-5.1.18.jar ./lib/

### 2.5、验证安装：

#查看sqoop的版本  
sqoop version

## 3.Sqoop命令执行

### 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  
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

### 3.2 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.

# #通过参数用下面查看数据库  
sqoop list-databases --connect jdbc:mysql://localhost:3306 --username root --password 123123;

上面命令中的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/config.conf,如下:

list-databases  
--connect  
jdbc:mysql://localhost:3306  
--username  
root  
--password  
123123

那么上面的执行的命令就可以变为:

bin/sqoop --options-file config.conf

为了让配置文件config.txt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  
123123

### 3.3 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:

#### 3.3.1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 |

##### 3.3.1.1 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:

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  
sqoop import --connect jdbc:mysql://localhost:3306/mysql --username root --password 123123;

##### 3.3.1.2 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  
bin/sqoop list-databases --connect jdbc:mysql://localhost:3306 --username root --password 123123;

##### 3.3.1.3 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  
bin/sqoop list-tables --connect jdbc:mysql://localhost:3306/mysql --username root --password 123123;

#### 3.3.2 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 | *n* 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) |

##### 3.3.2.1 指定表导入

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:

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp \  
--target-dir hdfs://miniwei:9000/sqoopdata/5  
--delete-target-dir

可以快速使用hdfs的命令查询结果

hdfs dfs -cat /sqoopdata/5/par\*

##### 3.3.2.2 指定列导入

如果想导入某几列,可以使用 --columns,如下:

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp \  
--columns 'empno,mgr' \  
--target-dir hdfs://miniwei:9000/sqoopdata/5 \  
--delete-target-dir

可以使用下面hdfs命令快速查看结果

hdfs dfs -cat /sqoopdata/5/par\*

##### 3.3.2.3 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:

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp \  
--columns 'empno,mgr' \  
--where 'empno>7800' \  
--target-dir hdfs://miniwei:9000/sqoopdata/5 \  
--delete-target-dir

用命令查询结果:

hdfs dfs -cat /sqoopdata/5/par\*

结果如下:

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

##### 3.3.2.4 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:

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--query 'select empno,mgr,job from emp WHERE empno>7800 and $CONDITIONS' \  
--target-dir hdfs://miniwei:9000/sqoopdata/5 \  
--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scoop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

查询执行结果

hdfs dfs -cat /sqoopdata/5/par\*

结果如下:

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

##### 3.3.2.5 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" \

##### 3.3.2.6 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:

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--query 'select empno,mgr,job from emp WHERE empno>7800 and $CONDITIONS' \  
--target-dir hdfs://miniwei:9000/sqoopdata/5 \  
--delete-target-dir \  
--split-by empno \  
-m 1

#### 3.3.3 导入到Hive中

##### 3.3.3.1 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。

##### 3.3.3.2 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 |

##### 3.3.3.3 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

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root \  
--password 123123 \  
--table emp \  
--delete-target-dir \  
--hive-import \  
--hive-overwrite \  
-m 1

在Hive中查看表:

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.Sqoop导入实战

### 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

准备数据:

create database comp;  
use comp;  
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);

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp -m 1

### 4.2 DBMS-HDFS

案例2：表没有主键，使用--split-by指定执行split的字段

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp \  
--split-by empno \  
--target-dir hdfs://miniwei:9000/sqoopdata/3

-- 出错  
Caused by: java.sql.SQLException: null, message from server: "Host 'hdp03' is not allowed to connect to this MySQL server"

解决方案：

先连接mysql：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)

GRANT ALL PRIVILEGES ON . TO 'root'@'%' IDENTIFIED BY 'mysql' WITH GRANT OPTION;  
 FLUSH PRIVILEGES;  
 grant all privileges on . to root@"localhost" identified by "mysql" with grant option;  
 FLUSH PRIVILEGES;

案例3：需要导入的数据不是全部的，而是带条件导入

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--table emp \  
--split-by empno \  
--where 'empno > 7777' \  
--target-dir hdfs://miniwei:9000/sqoopdata/5

案例4：要导入的数据，不想包含全部字段，只需要部分字段

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp \  
--username root --password 123123 \  
--split-by empno \  
--query 'select empno,ename,job from emp where empno > 7777 and $CONDITIONS' \  
--target-dir hdfs://miniwei:9000/sqoopdata/7

### 4.3 DBMS-Hive

案例5：将数据导入到hive中

bin/sqoop import --connect jdbc:mysql://localhost:3306/comp --username root --password 123123 --table emp --hive-import -m 1

### 4.4 增量导入数据

#### 4.4.1 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

#### 4.4.2 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

##### 4.4.2.1 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

vi ./ss.sh

写入以下内容:

#!/bin/bash   
#yesterday=`date -d "1 days ago" "+%Y-%m-%d"`   
yesterday='2019-01-01'   
sqoop import --connect jdbc:mysql://hadoop01:3306/sales\_source \   
--username root \   
--password root \   
--query "select \* from sales\_order where DATE(order\_date) = '${yesterday}' and \$CONDITIONS" \   
--driver com.mysql.jdbc.Driver \   
--delete-target-dir \   
--target-dir /user/hive/warehouse/sales\_ods.db/sales\_order/dt=${yesterday} \   
--split-by id \ -m 1 \   
--fields-terminated-by '\t' \   
--null-string '\\N' \  
--null-non-string '0

##### 4.4.2.2 increment的append方式：

#将会手动维护last-value   
sqoop import --connect jdbc:mysql://hadoop01:3306/sales\_source \   
--username root \   
--password root \   
--table sales\_order \   
--driver com.mysql.jdbc.Driver \  
 --target-dir /user/hive/warehouse/sales\_ods.db/sales\_order1/dt=2019-12-30 \   
--split-by id \   
 -m 1 \   
--check-column order\_number \   
--incremental append \  
--last-value 100000 \   
 --fields-terminated-by '\t' \   
 --null-string '\\N' \   
 --null-non-string '0'

## 5.Sqoop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：

bin/sqoop export \  
--connect jdbc:mysql://localhost:3306/comp \  
--username root \  
--password 123123 \  
--table emp \  
--export-dir hdfs://minihive:9000/sqoopdata/5

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
* 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。

## 6.Sqoop优化

### 6.1 -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vcore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。

### 6.2 --fetch-size n

一次取mysq1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-si ze能满足-一个mapper