

分布式ETL工具Sqoop

薛燊 星环高级数据工程师&AI工程师

2019年1月

目录 >

CONTENTS

- 1 Sqoop简介
- 2 Sqoop原理
- 3 Sqoop使用



1

chapter

Sqoop简介

- ✓ 什么是Sqoop
- ✓ Sqoop版本

- Sqoop项目始于2009年，早期为Hadoop的第三方模块，后来成为Apache的独立项目
- Sqoop是一个主要在Hadoop和关系数据库之间进行批量数据迁移的工具
 - Hadoop: HDFS、Hive、HBase、Inceptor、Hyperbase
 - 面向大数据集的批量导入导出
 - 将输入数据集分为N个切片，然后启动N个Map任务并行传输
 - 支持全量、增量两种传输方式
- 提供多种Sqoop连接器
 - 内置连接器
 - 经过优化的专用RDBMS连接器: MySQL、PostgreSQL、Oracle、DB2、SQL Server、Netzza等
 - 通用的JDBC连接器: 支持JDBC协议的数据库
 - 第三方连接器
 - 数据仓库: Teradata
 - NoSQL数据库: Couchbase

➤ Sqoop 1与 Sqoop 2的区别

- 二者完全不兼容，无法平滑升级

	Sqoop 1	Sqoop 2
版本号	1.4.x（1.4.7）	1.99.x（1.99.7）
系统架构	仅使用一个Sqoop客户端	引入Sqoop Server，集中管理连接器
访问方式	CLI	CLI、REST API、Java API、Web
安全机制	命令中包含用户名、密码	基于角色的安全机制
RDBMS连接器	经过优化的专用连接器，速度较快	通用的JDBC连接器，性能下降
RDBMS → Hive/HBase	直接导入	不直接导入，先存入HDFS
Hive/HBase → RDBMS	不直接导出，先存入HDFS	不直接导出，先存入HDFS

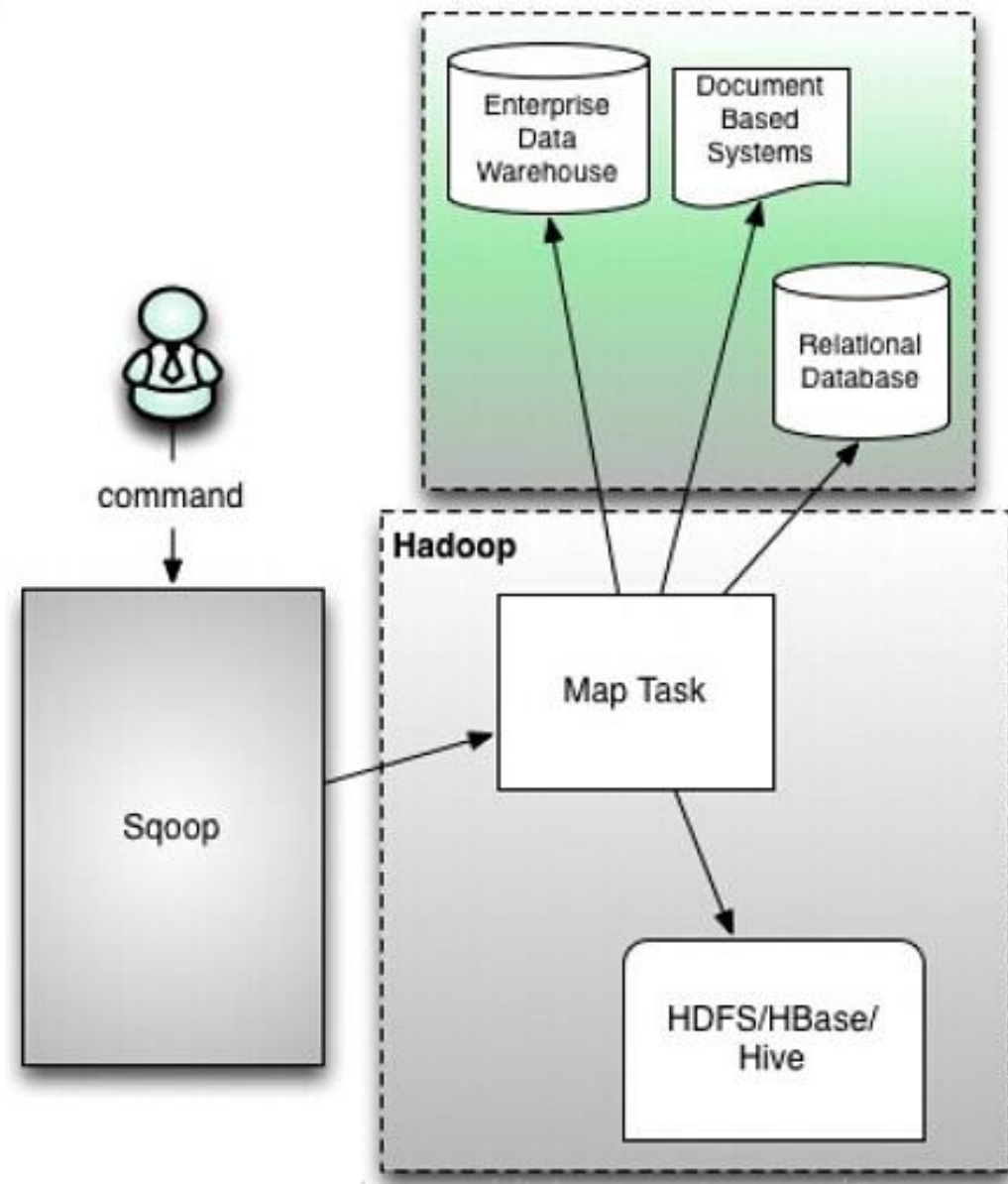
➤ Sqoop 1优缺点

- 优点

- 架构简单
- 部署简单
- 功能全面
- 稳定性较高
- 速度较快

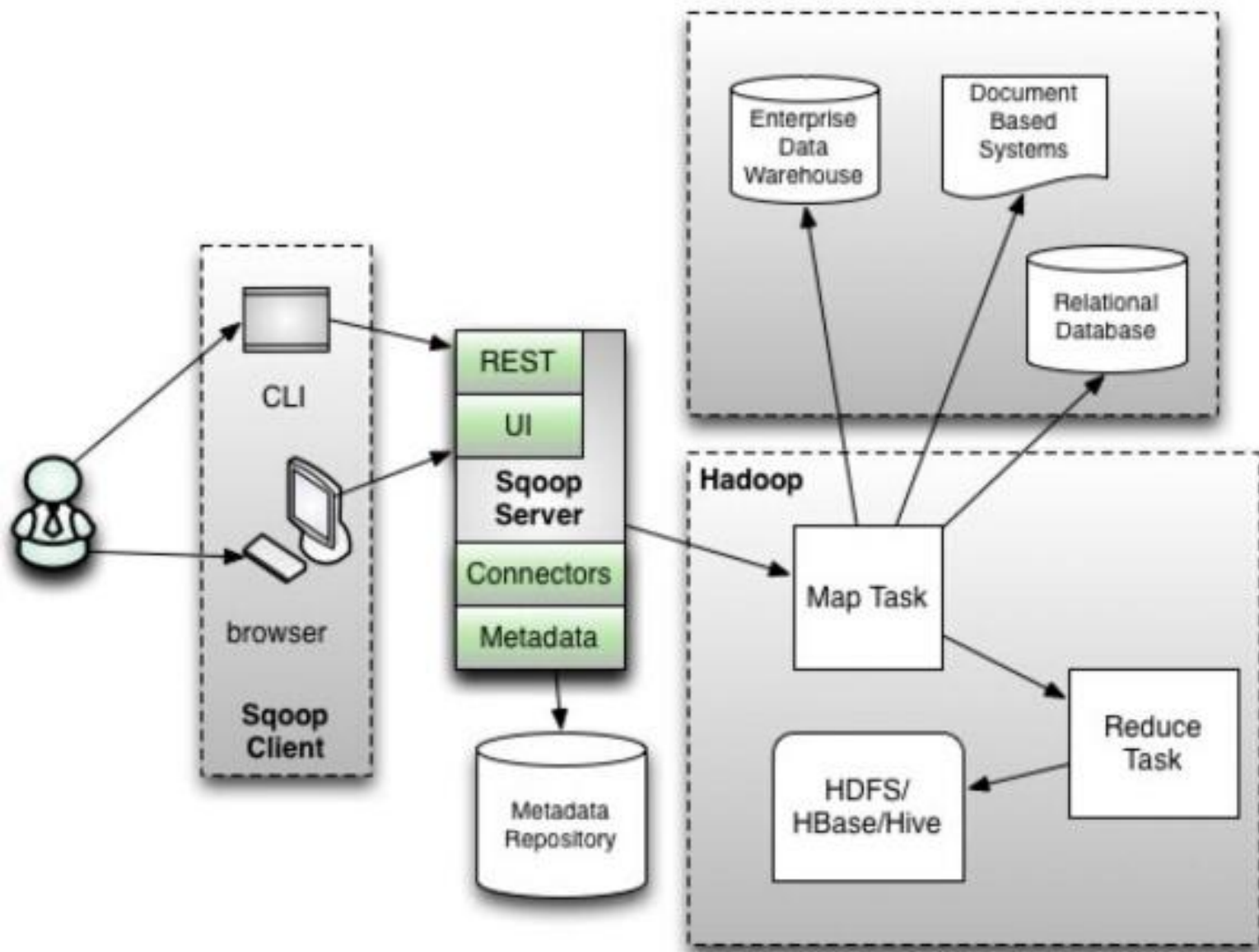
- 缺点


- 访问方式单一
- 命令行方式容易出错，格式紧耦合
- 安全机制不够完善，存在密码泄露风险



➤ Sqoop 2优缺点

- 优点
 - 访问方式多样
 - 集中管理连接器
 - 安全机制较完善
 - 支持多用户
- 缺点
 - 架构较复杂
 - 部署较繁琐
 - 稳定性一般
 - 速度一般



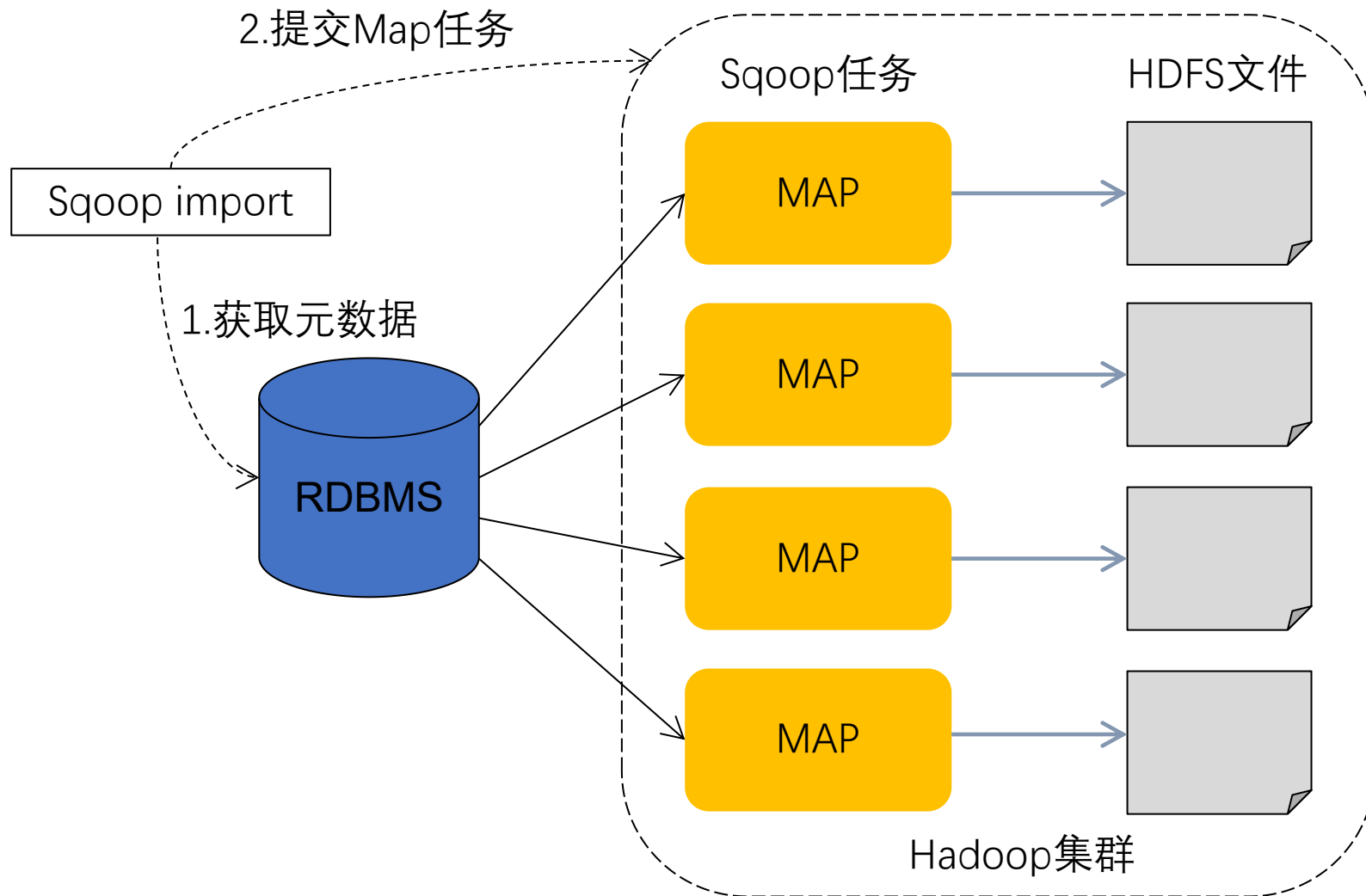


2 chapter

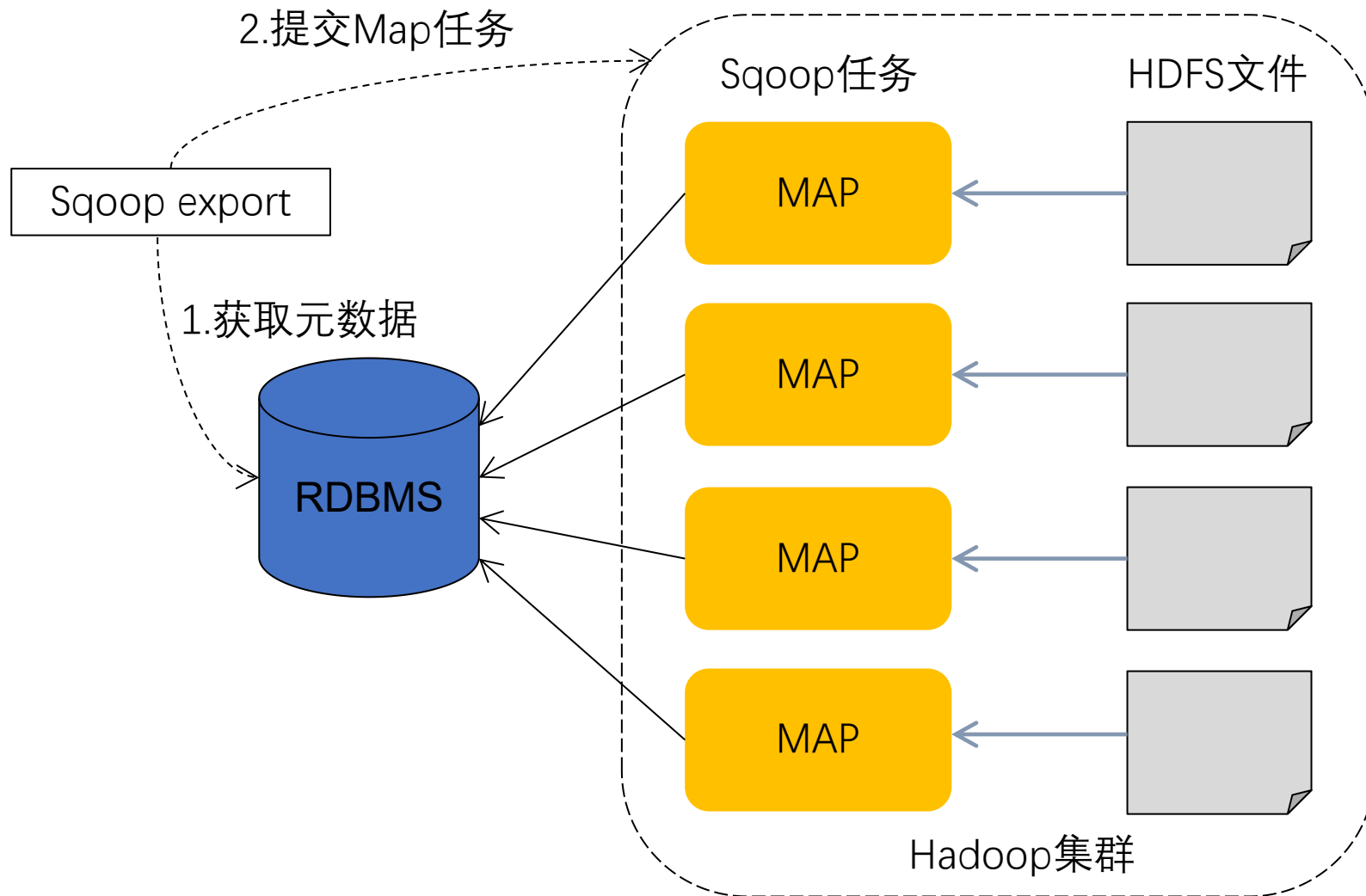
Sqoop原理


- ✓ 数据导入
- ✓ 数据导出

➤ RDBMS → Hadoop



➤ Hadoop → RDBMS





3 chapter

Sqoop使用

- ✓ 安装
- ✓ 基本用法
- ✓ 进阶用法

➤ 安装Sqoop

- 安装TDH Client（集成Sqoop）

- TDH Client下载：Transwarp Manager → 管理 → 下载客户端（tdh-client.tar），并解压
- TDH Client初始化：执行TDH Client目录下的init.sh脚本

- 安装开源Sqoop

- RedHat: yum install sqoop; Suse: zypper install sqoop

➤ 准备数据库驱动

- 将MySQL的JDBC驱动（mysql-connector-java-5.1.31.jar）拷贝到TDH Client目录下的sqoop/lib

➤ 测试

- 执行以下命令，若正常列出数据库，则说明Sqoop安装成功

```
/* 列出MySQL中的所有数据库 */  
# sqoop list-databases --connect jdbc:mysql://192.168.0.123:3316/ --username root --password transwarp
```


➤ Sqoop命令的两种形式

- 命令行
- 命令行 + 配置文件

```
/* 列出MySQL中的所有数据库 */  
# sqoop list-databases --connect jdbc:mysql://192.168.0.123:3316/ --username root --password transwarp  
# sqoop --options-file /users/homer/work/import.txt
```

```
/* 配置文件import.txt */  
list-databases  
--connect  
jdbc:mysql://192.168.0.123:3316/  
--username  
root  
--password  
transwarp
```

➤ 列出RDBMS的所有数据库

```
/* 明文密码 */
# sqoop list-databases \
  --connect jdbc:mysql://192.168.0.123:3316/\
  --username root \
  --password transwarp

/* 手工输入密码 */
# sqoop list-databases \
  --connect jdbc:mysql://192.168.0.123:3316/\
  --username root \
  -P

/* 密码文件 */
# sqoop list-databases \
  --connect jdbc:mysql://192.168.0.123:3316/\
  --username root \
  --password-file file:/root/pwd
```

Command Options	Description
list-databases	列出所有数据库
--connect	JDBC连接
--username	用户名
--password	明文密码
-P	手工输入密码
--password-file	密码文件（400权限）

➤ 列出数据库的所有表

```
/* 明文密码 */
# sqoop list-tables \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  --password transwarp

/* 手工输入密码 */
# sqoop list-tables \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  -P

/* 密码文件 */
# sqoop list-tables \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  --password-file file:/root/pwd
```

Command Options	Description
list-tables	列出所有表
--connect	JDBC连接
--username	用户名
--password	明文密码
-P	手工输入密码
--password-file	用户密码文件（400权限）

➤ 全量数据导入

```
/* 将数据全量导入HDFS */
# sqoop import \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  --password transwarp \
  --query "select * from t1 where $CONDITIONS" \
  --target-dir /user/root/person_all \
  --fields-terminated-by "\\01" \
  --hive-drop-import-delims \
  --null-string "\\N" \
  --null-non-string "\\N" \
  --split-by id \
  -m 6 \
```

Command Options	Description
import	数据导入
--query	SQL 查询语句
--target-dir	HDFS 目标目录（确保目录不存在，否则会报错）
--fields-terminated-by	列分隔符
--hive-drop-import-delims	删除数据中包含的Hive默认分隔符（^A, ^B, \n）
--null-string	string类型空值的替换符（Hive中Null用\n表示）
--null-non-string	非string类型空值的替换符
--split-by	数据切片字段（int类型，m>1时必须指定）
-m	Mapper任务数，默认为4

➤ 基于递增列的增量数据导入（Append方式）

```
/* 将递增列大于阈值的数据增量导入HDFS */
# sqoop import \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  --password transwarp \
  --query "select id, name from testdb where
    \${CONDITIONS}" \
  --target-dir /user/root/person_all \
  --split-by id \
  -m 6 \
  --incremental append \
  --check-column id \
  --last-value 4
```

Command Options	Description
--incremental append	基于递增列的增量导入（将递增列大于阈值的所有数据增量导入Hadoop）
--check-column	递增列（int）
--last-value	阈值（int）

➤ 基于时间列的增量数据导入（LastModified方式）

- 对比Append方式， LastModified方式可导入更新数据

```
/* 将时间列大于等于阈值的数据增量导入HDFS */
# sqoop import \
  --connect jdbc:mysql://192.168.0.123:3316/testdb \
  --username root \
  --password transwarp \
  --query "select id, name from testdb where \
    \$CONDITIONS" \
  --target-dir /user/root/person_all \
  --split-by id \
  -m 1 \
  --incremental lastmodified \
  --merge-key id \
  --check-column time \
  --last-value "2015-08-25 03:12:46"
```

Command Options	Description
--incremental lastmodified	基于时间列的增量导入（将时间列大于等于阈值的所有数据增量导入Hadoop）
--merge-key	合并列（主键，合并键值相同的记录）
--check-column	时间列（timestamp）
--last-value	阈值（timestamp）

➤ 从Oracle分区表中导入数据

```
/* 将Oracle分区表数据全量导入HDFS */  
# sqoop import  
--connect jdbc:oracle:thin:@$IP:$PORT/$SID \  
--username xxxx \  
--password xxxx \  
--query "select $COLUMNS from $ORACLE_USERNAME.$TABLENAME partition($PARTITION)  
where \"$CONDITIONS\" \  
--target-dir /data/sqoop/$USERNAME/$TABLENAME/$PARTITION \  
-m 1 \  
--fetch-size 10000 \  
--fields-terminated-by "\\01" \  
--hive-drop-import-delims \  
--null-string "\\N" \  
--null-non-string "\\N" \  

```

➤ 并发导入

- 通过-m参数， 设置多个Map任务， 实现数据并发导入
- -m大于1时， 必须设置--split-by， 并利用哈希取模实现数据均匀切片， 避免数据倾斜

RDBMS	Command Options
Oracle	-m 20 --split-by “MOD(ORA_HASH(col), 20)” --boundary-query “select 0, 19 from dual”
DB2	-m 20 --split-by “MOD(HASHEDVALUE(col), 20)” --boundary-query “VALUES (0, 19)”

➤ 并发度控制

- 数据导入的性能瓶颈
 - RDBMS和Hadoop集群的网络带宽
 - RDBMS的IO限制
- 数据导入的性能可按每个Map任务的处理速度5~10MB/s 做估算
- Map个数并非越多越好，过多的Map会导致RDBMS发生IO抢占，反而降低整体性能
- RDBMS的导出速度控制在60~80MB/s
- 实战中一般设置4~8个Mapper任务，通过Query人工均匀切分数据



Q&A

TRANSWARP
星环科技