Final Phantasy
DeLucia

SQOOP用户指南（V1.4.7）

Sqoop用户指南（v1.4.7）

1.简介

Sqoop是一种旨在在Hadoop与关系数据库或大型机之间传输数据的工具。您可以使用Sqoop将数据从MySQL或Oracle等关系数据库管理系统（RDBMS）或大型机导入Hadoop分布式文件系统（HDFS），在Hadoop MapReduce中转换数据，然后将数据导出回RDBMS 。

Sqoop依靠数据库描述要导入的数据的模式来自动执行此过程的大部分时间。Sqoop使用MapReduce导入和导出数据，这提供了并行操作以及容错能力。

本文档介绍了如何开始使用Sqoop在数据库和Hadoop或大型机之间迁移数据到Hadoop，并提供了有关Sqoop命令行工具套件操作的参考信息。本文档适用于：

- 系统和应用程序程序员
- 系统管理员
- 数据库管理员
- 数据分析师
- 数据工程师

2.支持的版本

本文档适用于Sqoop v1.4.7。

3. Sqoop版本

Sqoop是Apache Software Foundation的开源软件产品。

Sqoop的软件开发位于http://sqoop.apache.org。 在该站点上，您可以获得：

- Sqoop的新版本及其最新源代码

- 问题追踪器
- 包含Sqoop文档的Wiki

4.先决条件

该产品需要以下先决条件知识：

- 基本计算机技术和术语
- 熟悉命令行界面，例如 bash
- 关系数据库管理系统
- 基本了解Hadoop的用途和操作

在使用Sqoop之前，必须先安装和配置Hadoop发行版。Sqoop当前支持4个主要的Hadoop版本-0.20、0.23、1.0和2.0。

本文档假定您使用的是Linux或类似Linux的环境。如果使用Windows，则可以使用cygwin完成以下大多数任务。如果您使用的是Mac OS X，应该会看到很少的兼容性错误（如果有的话）。Sqoop主要在Linux上进行操作和测试。

5.基本用法

使用Sqoop，您可以*将*数据从关系数据库系统或大型机导入HDFS。导入过程的输入是数据库表或大型机数据集。对于数据库，Sqoop会将表逐行读取到HDFS中。对于大型机数据集，Sqoop将从每个大型机数据集中读取记录到HDFS中。此导入过程的输出是一组文件，其中包含导入的表或数据集的副本。导入过程是并行执行的。因此，输出将在多个文件中。这些文件可以是定界的文本文件（例如，用逗号或制表符分隔每个字段），或包含序列化记录数据的二进制Avro或SequenceFiles。

导入过程的副产品是生成的Java类，它可以封装导入表的一行。此类在Sqoop本身的导入过程中使用。还为您提供了此类的Java源代码，以用于后续的MapReduce数据处理中。此类可以将数据与SequenceFile格式进行序列化和反序列化。它还可以解析记录的分隔文本形式。这些功能使您可以快速开发在处理管道中使用HDFS存储的记录的MapReduce应用程序。您还可以使用自己喜欢的任何其他工具自由解析定界记录数据。

在处理了导入的记录（例如，使用MapReduce或Hive）之后，您可能会有一个结果数据集，然后可以将其*导出*回关系数据库。Sqoop的导出过程将从HDFS并行读取一组定界的文本文件，将它们解析为记录，并将它们作为新行插入目标数据库表中，以供外部应用程序或用户使用。

Sqoop包括一些其他命令，这些命令使您可以检查正在使用的数据库。例如，您可以列出可用的数据库模式（使用 sqoop-list-databases 工具）和模式中的表（使用 sqoop-list-tables 工具）。Sqoop还包括原始SQL执行外壳（该 sqoop-eval 工具）。

导入，代码生成和导出过程的大多数方面都可以自定义。对于数据库，您可以控制导入的特定行范围或列。您可以为数据的基于文件的表示形式以及所使用的文件格式指定特定的分隔符和转义符。您还可以控制在生成的代码中使用的类或程序包名称。本文档的后续部分介绍了如何为Sqoop指定这些参数和其他参数。

6. Sqoop工具

Sqoop是相关工具的集合。要使用Sqoop，请指定要使用的工具以及控制该工具的参数。

如果Sqoop是从其自己的源代码编译的，则可以通过运行该 bin/sqoop 程序来运行Sqoop，而无需进行正式的安装过程。Sqoop打包部署（例`如Apache Bigtop附带的RPM）的用户将看到此程序安装为 /usr/bin/sqoop 。本文档的其余部分将该程序称为 sqoop 。例如：

```
$ sqoop工具名称[工具参数]
```

以下以 $ 字符开头的示例表明必须在终端提示符下输入命令（例如 bash ）。的 $ 字符表示提示本身;您不应通过键入来启动这些命令 $ 。您也可以在段落文本中内联输入命令；例如， sqoop help 。这些示例没有显示 $ 前缀，但是您应该以相同的方式输入它们。不要将 $ 示例中的shell提示与 $ 环境变量名称之前的混

淆。例如，字符串文字$HADOOP_HOME 包括“ $ ”。

Sqoop附带了一个帮助工具。要显示所有可用工具的列表，请键入以下命令：

```
$ sqoop帮助
用法：sqoop COMMAND [ARGS]

可用命令：
  codegen生成代码以与数据库记录进行交互
  create-hive-table将表定义导入Hive
  评估SQL语句并显示结果
  导出将HDFS目录导出到数据库表
  帮助列出可用的命令
  导入将表从数据库导入到HDFS
  import-all-tables从数据库导入表到HDFS
  import-mainframe将大型机数据集导入HDFS
  list-databases列出服务器上的可用数据库
  list-tables列出数据库中的可用表
  版本显示版本信息

有关特定命令的信息，请参见“ sqoop帮助命令”。
```

您可以显示通过输入特定工具帮助： sqoop help (tool-name) ；例如， sqoop help import 。

您还可以将 --help 参数添加到任何命令：中 sqoop import --help 。

6.1。使用命令别名

除了键入 sqoop (toolname) 语法，您还可以使用指定 sqoop-(toolname) 语法的别名脚本。例如，脚本 sqoop-import ， sqoop-export 等等每个选择一个特定的工具。

6.2。控制Hadoop安装

您可以通过Hadoop提供的程序启动功能来调用Sqoop。在 sqoop 命令行程序是运行在一个包装 bin/hadoop 附带的Hadoop脚本。如果您的计算机上有多个Hadoop安装，则可以通过设置 $HADOOP_COMMON_HOME 和 $HADOOP_MAPRED_HOME 环境变量来选择Hadoop安装。

例如：

```
$ HADOOP_COMMON_HOME = /路径/到/某些/ hadoop \
  HADOOP_MAPRED_HOME = /路径/到/一些/ hadoop-mapreduce \
  sqoop导入-参数...
```

或者：

```
$ export HADOOP_COMMON_HOME = / some / path / to / hadoop
$ export HADOOP_MAPRED_HOME = / some / path / to / hadoop-mapreduce
$ sqoop import-参数...
```

如果未设置这些变量中的任何一个，则Sqoop将退回到 $HADOOP_HOME 。如果两者均未设置，则Sqoop将分别使用Apache Bigtop /usr/lib/hadoop 和 的默认安装位置 /usr/lib/hadoop-mapreduce 。

$HADOOP_HOME/conf/ 除非 $HADOOP_CONF_DIR 设置了环境变量，否则将从加载活动的Hadoop配置。

6.3。使用泛型和特定参数

要控制每个Sqoop工具的操作，请使用通用参数和特定参数。

例如：

```
$ sqoop帮助导入
用法：sqoop导入[GENERIC-ARGS] [TOOL-ARGS]
```

```
常用参数：
   --connect <jdbc-uri>指定JDBC连接字符串
   --connect-manager <类名称>指定要使用的连接管理器类
   --driver <class-name>手动指定要使用的JDBC驱动程序类
   --hadoop-mapred-home <dir>覆盖$ HADOOP_MAPRED_HOME
   --help打印用法说明
   --password-file设置包含认证密码的文件的路径
   -P从控制台读取密码
   --password <密码>设置身份验证密码
   --username <用户名>设置身份验证用户名
   --verbose在工作时打印更多信息
   --hadoop-home <dir>已弃用。覆写$ HADOOP_HOME

[...]

通用Hadoop命令行参数：
（必须在任何特定于工具的参数之前）
支持的通用选项是
-conf <配置文件>指定应用程序配置文件
-D <property = value>使用给定属性的值
-fs <local | namenode：port>指定一个名称节点
-jt <local | jobtracker：port>指定作业跟踪器
-files <以逗号分隔的文件列表>指定要以逗号分隔的文件，将其复制到map reduce集群中
-libjars <以逗号分隔的jar列表>指定以逗号分隔的jar文件，以包括在类路径中。
-archives <以逗号分隔的存档列表>指定要在计算机上取消存档的以逗号分隔的存档。

常规命令行语法为
bin / hadoop命令[genericOptions] [commandOptions]
```

您必须在工具名称之后但**在**任何特定**于**工具的参数（例如 ）**之前**提供通用参数 -conf， -D 等等。请注意，一般的Hadoop参数以一个破折号（）开头，而特定于工具的参数以两个破折号（）开头，除非它们是单个字符（例如）。 --connect - -- -P

的 -conf， -D， -fs 和 -jt 参数控制配置和Hadoop服务器设置。例如， -D mapred.job.name=<job_name> 可以使用设置Sqoop启动的MR作业的名称，如果未指定，则该名称默认为该作业的jar名称-从使用的表名称派生。

的 -files， -libjars 和 -archives 参数通常不使用Sqoop，但它们包含作为Hadoop的内部争论，分析系统的一部分。

6.4。使用选项文件传递参数

使用Sqoop时，为了方便起见，可以将未在调用之间更改的命令行选项放在选项文件中。选项文件是一个文本文件，其中每一行都按照在命令行中否则会出现的顺序来标识一个选项。选项文件允许通过在中间行的末尾使用反斜杠字符来在多行中指定单个选项。还支持选项文件中以哈希字符开头的注释。注释必须在新行上指定，并且不得与选项文本混合。扩展选项文件时，将忽略所有注释和空行。除非选项显示为带引号的字符串，否则将忽略任何前导或尾随空格。带引号的字符串（如果使用的话）不能超出其指定行的范围。

可以在命令行中的任何位置指定选项文件，只要它们中的选项遵循其他规定的选项排序规则即可。例如，无论从何处加载选项，它们都必须遵循以下顺序：首先显示通用选项，然后显示特定于工具的选项，最后是打算传递给子程序的选项。

要指定选项文件，只需在方便的位置创建一个选项文件，然后将其通过 --options-file 参数传递给命令行 。

每当指定选项文件时，在调用该工具之前都会在命令行上对其进行扩展。如果需要，您可以在同一调用中指定多个选项文件。

例如，可以选择指定以下导入的Sqoop调用，如下所示：

```
$ sqoop import --connect jdbc：mysql：// localhost / db --username foo --table TEST

$ sqoop --options-file /users/homer/work/import.txt --table TEST
```

选项文件 /users/homer/work/import.txt 包含以下内容：

```
进口
- 连接
jdbc：mysql：// localhost / db
- 用户名
富
```

选项文件可以包含空行和注释，以提高可读性。因此，如果选项文件 /users/homer/work/import.txt 包含以下内容，则上面的示例将完全相同：

```
#
#Sqoop导入的选项文件
#

#指定正在调用的工具
进口
```

```
#连接参数和值
- 连接
jdbc：mysql：// localhost / db

#用户名参数和值
- 用户名
富

#
#其余选项应在命令行中指定。
#
```

6.5。使用工具

以下各节将介绍每种工具的操作。这些工具以最有可能的顺序列出，您会发现它们很有用。

7 sqoop-import

7.1。目的

该 import 工具将单个表从RDBMS导入到HDFS。表中的每一行在HDFS中均表示为单独的记录。记录可以存储为文本文件（每行一个记录），或以二进制表示形式存储为Avro或SequenceFiles。

7.2。句法

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

尽管Hadoop通用参数必须在任何导入参数之前，但是您可以相对于彼此以任何顺序键入导入参数。

在本文档中，参数按功能分组到集合中。一些工具中提供了一些集合（例如，“ common”自变量）。仅在本文档的第一个介绍中给出了对它们功能的扩展描述。

表1.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

7.2.1。连接到数据库服务器

Sqoop旨在将表从数据库导入HDFS。为此，您必须指定描述如何连接到数据库的*连接字符串*。该*连接字符串*是类似于URL，并传达给Sqoop与 --connect 争论。这描述了要连接的服务器和数据库；它也可以指定端口。例如：

```
$ sqoop import --connect jdbc:mysql://database.example.com/employees
```

该字符串将连接到 employees 主机上命名的MySQL数据库 database.example.com 。如果要在分布式Hadoop群集中使用Sqoop，**请不要**使用URL ，这一点很重要 localhost 。您提供的连接字符串将在整个MapReduce集群的TaskTracker节点上使用；如果您指定文字名称 localhost ，则每个节点将连接到一个不同的数据库（或者更可能根本没有数据库）。相反，您应该使用所有远程节点都可以看到的数据库主机的完整主机名或IP地址。

您可能需要先对数据库进行身份验证，然后才能访问它。您可以使用 --username 来向数据库提供用户名。Sqoop提供了几种不同的方法来向数据库提供密码（安全和不安全），这将在下面详细介绍。

向数据库提供密码的安全方式。 您应该将密码保存在用户主目录中的文件中，该文件具有400个权限，并使用 **--password-file** 参数指定该文件的路径，这是输入凭据的首选方法。然后，Sqoop将使用安全方式从文件中读取密码，并将其传递给MapReduce集群，而不会在作业配置中公开密码。包含密码的文件可以在本地FS或HDFS上。例如：

```
$ sqoop import --connect jdbc:mysql://database.example.com/employees \
    --username venkatesh-密码文件${user.home}/。password
```

警告 Sqoop将读取密码文件的全部内容，并将其用作密码。这将包括大多数文本编辑器默认添加的任何尾随空格字符，例如换行符。您需要确保密码文件仅包含属于密码的字符。在命令行上，可以将 echo 带开关的命令与命令一起使用， -n 以存储密码，且不包含任何尾随空格。例如，要存储密码， secret 您可以致电 echo -n "secret" > password.file 。

提供密码的另一种方法是使用 -P 参数，该参数将从控制台提示中读取密码。

保护密码不被窃取。 Hadoop 2.6.0提供了一个API，用于将密码存储与应用程序分开。此API称为凭据提供的API，并且提供了一个新的 credential 命令行工具来管理密码及其别名。密码及其别名存储在受密码保护的密钥库中。密钥库密码可以通过环境变量提供给命令行上的密码提示，也可以默认为软件定义的常量。请检查Hadoop文档中有关此功能的用法。

使用凭据提供程序工具存储密码并适当更新Hadoop配置后，所有应用程序都可以选择使用别名代替实际密码，并在运行时解析要使用的密码的别名。

由于用于存储凭据提供程序的密钥库或类似技术是在各个组件之间共享的，因此可以将各种应用程序，各种数据库和其他密码的密码安全地存储在它们中，并且只需要在配置文件中公开别名，以防止密码被泄露。可见的。

如果正在使用的基础Hadoop版本中提供了此功能，则Sqoop进行了增强以允许使用此功能。引入了一个新选项来在命令行上提供别名，而不是实际密码（–password-alias）。此选项的参数值是与实际密码关联的存储器上的别名。用法示例如下：

```
$ sqoop import --connect jdbc：mysql：//database.example.com/employees \
    --username dbuser --password-alias mydb.password.alias
```

同样，如果命令行选项不是首选选项，则别名可以保存在–password-file选项随附的文件中。与此同时，应将Sqoop配置参数org.apache.sqoop.credentials.loader.class设置为提供别名解析的类名： org.apache.sqoop.util.password.CredentialProviderPasswordLoader

用法示例如下（假设.password.alias具有真实密码的别名）：

```
$ sqoop import --connect jdbc：mysql：//database.example.com/employees \
    --username dbuser --password-file $ {user.home} /。password-alias
```

警告 该 --password 参数不安全，因为其他用户可能会通过诸如的程序输出从命令行参数读取您的密码 ps 。与使用 **-P** 参数相比，参数是首选方法 --password 。凭证仍可以使用不安全手段在MapReduce群集的节点之间传输。例如：

```
$ sqoop import --connect jdbc：mysql：//database.example.com/employees \
    --username aaron-密码12345
```

Sqoop自动支持多个数据库，包括MySQL。 jdbc:mysql:// Sqoop中会自动处理以开头的连接字符串。 （“受支持的数据库”部分提供了具有内置支持的数据库的完整列表。对于某些数据库，您可能需要自己安装JDBC驱动程序。）

您可以将Sqoop与其他任何符合JDBC的数据库一起使用。首先，为要导入的数据库类型下载适当的JDBC驱动程序，然后将.jar文件安装在 $SQOOP_HOME/lib 客户端计算机上的目录中。 （ /usr/lib/sqoop/lib 如果是从RPM或Debian软件包安装的，则将是这样。）每个驱动程

序 .jar 文件还具有一个特定的驱动程序类，该类定义了驱动程序的入口点。例如，MySQL的Connector / J库的驱动程序类为 com.mysql.jdbc.Driver 。请参阅您的数据库供应商特定的文档，以确定主驱动程序类。此类必须作为Sqoop的参数提供 --driver 。

例如，要连接到SQLServer数据库，请首先从microsoft.com下载驱动程序并将其安装在Sqoop库路径中。

然后运行Sqoop。例如：

```
$ sqoop import --driver com.microsoft.jdbc.sqlserver.SQLServerDriver \
    --connect <连接字符串> ...
```

使用JDBC连接到数据库时，可以选择使用属性文件通过属性文件指定其他JDBC参数 --connection-param-file 。该文件的内容被解析为标准Java属性，并在创建连接时传递到驱动程序中。

笔记 通过可选属性文件指定的参数仅适用于JDBC连接。使用除JDBC以外的其他连接的任何快速路径连接器都将忽略这些参数。

表2.验证参数更多详细信息

争论	描述
--validate	启用对复制数据的验证，仅支持单表复制。
--validator <class-name>	指定要使用的验证器类。
--validation-threshold <class-name>	指定要使用的验证阈值类。
--validation-failurehandler <class-name>	指定要使用的验证失败处理程序类。

表3.导入控制参数：

争论	描述

争论	描述
--append	将数据追加到HDFS中的现有数据集
--as-avrodatafile	将数据导入Avro数据文件
--as-sequencefile	将数据导入到SequenceFiles
--as-textfile	以纯文本格式导入数据（默认）
--as-parquetfile	将数据导入Parquet文件
--boundary-query <statement>	用于创建拆分的边界查询
--columns <col,col,col…>	要从表中导入的列
--delete-target-dir	删除导入目标目录（如果存在）
--direct	如果数据库存在，则使用直接连接器
--fetch-size <n>	一次要从数据库读取的条目数。
--inline-lob-limit <n>	设置内联LOB的最大大小
-m,--num-mappers <n>	使用*n*个地图任务并行导入
-e,--query <statement>	导入的结果 *statement* 。
--split-by <column-name>	该表的列用于拆分工作单位。不能与 --autoreset-to-one-mapper 选项一起使用 。
--split-limit <n>	每个分割尺寸的上限。这仅适用于Integer和Date列。对于日期或时间戳记字段，以秒为单位进行计算。
--autoreset-to-one-mapper	如果表没有主键并且没有提供分隔列，则导入应使用一个映射器。不能与 --split-by <col> 选项一起使用 。
--table <table-name>	表格阅读
--target-dir <dir>	HDFS目标目录
--temporary-rootdir <dir>	导入期间创建的临时文件的HDFS目录（覆盖默认的“ _sqoop”）
--warehouse-dir <dir>	表目标的HDFS父级
--where <where clause>	导入期间要使用的WHERE子句
-z,--compress	启用压缩

争论	描述
--compression-codec <c>	使用Hadoop编解码器（默认gzip）
--null-string <null-string>	要为字符串列的空值写入的字符串
--null-non-string <null-string>	非字符串列要为空值写入的字符串

在 --null-string 和 --null-non-string 参数都是可选的。\如果未指定，那么字符串“null”将被使用。

7.2.2。选择要导入的数据

Sqoop通常以表为中心的方式导入数据。使用 --table 参数选择要导入的表。例如， --table employees 。此参数还可以标识数据库中的一个 VIEW 或其他类似表的实体。

默认情况下，将选择表中的所有列进行导入。导入的数据以其“自然顺序”写入HDFS；也就是说，包含A，B和C列的表将导致导入数据，例如：

```
A1，B1，C1
A2，B2，C2
...
```

您可以选择一个列子集，并通过使用 --columns 参数来控制列的顺序。这应该包括要导入的以逗号分隔的列列表。例如： --columns "name,employee_id,jobtitle" 。

您可以通过 WHERE 在import语句中添加SQL子句来控制要导入的行。默认情况下，Sqoop生成形式为的语句 SELECT <column list> FROM <table name> 。您可以 WHERE 在该 --where 参数后附加一个 子句。例如： --where "id > 400" 。仅 id 将导入列值大于400的行。

默认情况下，sqoop将使用查询 select min(<split-by>), max(<split-by>) from <table name> 来查找创建拆分的边界。在某些情况下，该查询不是最佳查询，因此您可以使用 --boundary-query 参数指定返回两个数字列的任意查询。

7.2.3。自由格式查询导入

Sqoop还可以导入任意SQL查询的结果集。除了使用的 --table ， --columns 和 --where 参数，您可以指定与SQL语句 --query 的参数。

导入自由格式查询时，必须使用来指定目标目录 --target-dir 。

如果要并行导入查询的结果，则每个地图任务都需要执行查询的副本，其结果将由Sqoop推断出的边界条件进行划分。您的查询必须包含 $CONDITIONS 每个Sqoop进程将用唯一条件表达式替换的令牌。您还必须使用来选择一个拆分列 --split-by 。

例如：

```
$ sqoop import \
  --query'SELECT a。*，b。*从（a.id == b.id）WHERE $ CONDITIONS上的JOIN b'
  --split-by a.id --target-dir / user / foo / joinresults
```

另外，通过指定一个映射任务，查询可以一次执行并依次导入 -m 1 ：

```
$ sqoop import \
  --query'SELECT a。*，b。*从（a.id == b.id）WHERE $ CONDITIONS上的JOIN b'
  -m 1 --target-dir / user / foo / joinresults
```

笔记 如果要发布用双引号（“）括起来的查询，则您将不得不使用 \$CONDITIONS 而不是仅仅 $CONDITIONS ，例如： "SELECT * FROM x WHERE a='foo' AND \$CONDITIONS"

笔记 在当前版本的Sqoop中使用自由格式查询的功能仅限于简单查询，在该查询中子句中没有模棱两可的投影且没有任何 OR 条件 WHERE 。使用复杂的查询（例如具有子查询或联接的查询导致模棱两可的投影）可能会导致意外的结果。

7.2.4。控制并行

Sqoop从大多数数据库源并行导入数据。您可以指定（并行处理）的地图任务的数量使用通过执行进口 -m 或 --num-mappers 争论。这些参数中的每个参数取一个整数值，该整数值与要采用的并行度相对应。默认情况下，使用四个任务。通过将此值增加到8或16，某些数据库可能会看到性能提高。不要将并行度提高到比MapReduce集群中可用的并行度大的水平；任务将连续运行，并且可能会增加执行导入所需的时间。同样，不要将并行度的增加幅度提高到数据库可以合理支持的程度。将100个并发客户端连接到数据库可能会增加数据库服务器的负载，从而导致性能下降。

在执行并行导入时，Sqoop需要一个可用来划分工作负载的标准。Sqoop使用*拆分列*拆分工作负载。默认情况下，Sqoop将识别表中的主键列（如果存在）并将其用作拆分列。从数据库中检索拆分列的低值和高值，并且映射任务对总范围的大小均匀的分量进行操作。例如，如果您有一个表的主键列的 id 最小值为0，最大值为1000，并且Sqoop被定向为使用4个任务，则Sqoop将运行四个进程，每个进程执行 SELECT * FROM sometable WHERE id >= lo AND id < hi 带有 (lo, hi) set形式的SQL语句。到（0，250），（250，500），（500，750）和（750，1001）中。

如果主键的实际值在其范围内分布不均匀，则可能导致任务不平衡。您应该使用该 --split-by 参数显式选择其他列。例如， --split-by employee_id 。Sqoop当前无法在多列索引上拆分。如果您的表没有索引列，或者具有多列键，那么您还必须手动选择一个拆分列。

用户可以 --num-mapers 使用using --split-limit 选项覆盖。使用该 --split-limit 参数对创建的分割部分的大小施加限制。如果创建的拆分的大小大于此参数中指定的大小，则将调整拆分的大小以适合此限制，并且拆分的数量将随之更改，这会影响实际的映射器数量。如果根据提供的 --num-mappers 参数计算的分割大小超过 --split-limit 参数，则实际的映射器数将增加。如果参数中指定的值为 --split-limit 0或负数，则将完全忽略该参数，并且将根据分割数来计算分割大小映射器。

如果表未定义主键 --split-by <col> 且未提供，则导入将失败，除非使用该 --num-mappers 1 选项将映射器的数量显式设置为1或使用该 --autoreset-to-one-mapper 选项。该选项 --autoreset-to-one-mapper 通常与import-all-tables工具一起使用，以自动处理没有模式中主键的表。

7.2.5。控制分布式缓存

每次启动Sqoop作业时，Sqoop都会将$ SQOOP_HOME / lib文件夹中的jar复制到作业缓存中。由Oozie启动时，这是不必要的，因为Oozie使用自己的Sqoop共享库，该库将Sqoop依赖项保留在分布式缓存中。Oozie将在第一个Sqoop作业期间仅在每个工作节点上对Sqoop依赖项进行本地化，然后将工作节点上的jar重新用于次要工作。 --skip-dist-cache 由Oozie启动时，在Sqoop命令中使用option将跳过Sqoop将其依赖项复制到作业缓存并节省大量I / O的步骤。

7.2.6。控制导入过程

默认情况下，导入过程将使用JDBC，JDBC提供了合理的跨供应商导入渠道。通过使用特定于数据库的数据移动工具，某些数据库可以以更高性能的方式执行导入。例如，MySQL提供了 mysqldump 可以非常快速地将数据从MySQL导出到其他系统的工具。通过提供 --direct 参数，可以指定Sqoop应该尝试直接导入通道。与使用JDBC相比，此通道的性能可能更高。

有关在每个特定RDBMS上使用直接模式的详细信息，安装要求，可用的选项和限制，请参见第25节“特定连接器的注意事项”。

默认情况下，Sqoop将导入一个表，该表命名 foo 为 foo HDFS中主目录内的目录。例如，如果您的用户名是 someuser ，则导入工具将写入 /user/someuser/foo/(files) 。您可以使用 --warehouse-dir 参数调整导入的父目录。例如：

```
$ sqoop import --connnect <connect-str> --table foo --warehouse-dir / shared \
    ...
```

该命令将写入目录中的一组文件 /shared/foo/ 。

您还可以显式选择目标目录，如下所示：

```
$ sqoop import --connnect <connect-str> --table foo --target-dir / dest \
    ...
```

这会将文件导入 /dest 目录。 --target-dir 与不兼容 --warehouse-dir 。

使用直接模式时，可以指定应传递给基础工具的其他参数。如果参数 -- 是在命令行上给出的，则随后的参数将直接发送到基础工具。例如，以下内容可调整所使用的字符集 mysqldump ：

```
$ sqoop import --connect jdbc:mysql://server.foo.com/db --table bar \
    --direct---default-character-set = latin1
```

默认情况下，导入将转到新的目标位置。如果目标目录已存在于HDFS中，则Sqoop将拒绝导入并覆盖该目录的内容。如果使用该 --append 参数，则Sqoop会将数据导入到临时目录，然后以与该目录中现有文件名不冲突的方式将文件重命名为普通目标目录。

7.2.7。控制事务隔离

默认情况下，Sqoop使用映射器中的已读提交事务隔离来导入数据。这可能不是所有ETL工作流程中的理想选择，并且可能希望减少隔离保证。该 --relaxed-isolation 选项可用于指示Sqoop使用读取未提交的隔离级别。

该 read-uncommitted 隔离级别不支持所有数据库（例如Oracle），因此指定选项 --relaxed-isolation 可能无法在所有数据库的支持。

7.2.8。控制类型映射

Sqoop已预先配置为将大多数SQL类型映射到适当的Java或Hive代表。但是，默认映射可能并不适合所有人，并且可能会被 --map-column-java （用于将映射更改为Java）或 --map-column-hive （用于更改Hive映射）所覆盖。

表4.覆盖映射的参数

争论	描述
--map-column-java <mapping>	覆盖从SQL到Java类型的已配置列的映射。
--map-column-hive <mapping>	覆盖已配置列的从SQL到Hive类型的映射。

Sqoop希望以逗号分隔的映射列表形式为<列名> = <新类型>。例如：

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

请注意，在–map-column-hive选项中指定逗号，应使用URL编码的键和值，例如，使用DECIMAL（1%2C%201）代替DECIMAL（1、1）。

如果不使用某些已配置的映射，则Sqoop将出现异常。

7.2.9。模式名称处理

当sqoop从企业存储中导入数据时，表名和列名可能包含无效的Java标识符字符或Avro / Parquet标识符字符。为了解决这个问题，在架构创建过程中，sqoop将这些字符转换为_。以_（下划线）字符开头的任何列名称都将转换为具有两个下划线字符。例如，_AVRO将被转换为__AVRO。

对于HCatalog导入，当映射到HCatalog列时，列名将转换为小写。将来可能会改变。

7.2.10。增量进口

Sqoop提供了一种增量式导入模式，该模式可用于仅检索比某些先前导入的行集新的行。

以下参数控制增量导入：

表5.增量导入参数：

争论	描述
--check-column (col)	指定在确定要导入的行时要检查的列。（该列的类型不应为CHAR / NCHAR / VARCHAR / VARNCHAR / LONGVARCHAR / LONGNVARCHAR）

争论	描述
--incremental (mode)	指定Sqoop如何确定哪些行是新的。 mode include append 和的法律价值 lastmodified 。
--last-value (value)	指定上一次导入中检查列的最大值。

Sqoop支持两种类型的增量导入： append 和 lastmodified 。您可以使用 --incremental 参数指定要执行的增量导入的类型。

append 导入表时，应指定模式，在该表中，随着行ID值的增加，将不断添加新行。您可以使用指定包含行ID的列 --check-column 。Sqoop导入行，其中check列的值大于用所指定的值 --last-value 。

Sqoop支持的另一种表更新策略称为 lastmodified 模式。当源表的行可能会更新时，应该使用此选项，并且每次此类更新会将上次修改的列的值设置为当前时间戳。 --last-value 导入检查列保存的时间戳比使用指定的时间戳更新的时间戳的行。

在增量导入结束时，应 --last-value 为后续导入指定的值 将显示在屏幕上。运行后续导入时，应 --last-value 以这种方式指定以确保仅导入新数据或更新数据。通过将增量导入创建为已保存的作业来自动处理此问题，这是执行重复增量导入的首选机制。有关更多信息，请参见本文档后面有关已保存作业的部分。

7.2.11。档案格式

您可以采用以下两种文件格式之一导入数据：定界文本或SequenceFiles。

带分隔符的文本是默认的导入格式。您也可以使用 --as-textfile 参数明确指定它。此参数会将每个记录的基于字符串的表示形式写入输出文件，并在各个列和行之间使用定界符。这些定界符可以是逗号，制表符或其他字符。（可以选择定界符；请参阅“输出行格式参数”。）以下是示例基于文本的导入的结果：

```
1,here is a message,2010-05-01
```

```
2,happy new year!,2010-01-01
3,another message,2009-11-12
```

带分隔符的文本适用于大多数非二进制数据类型。它还很容易支持其他工具（例如Hive）的进一步操作。

SequenceFiles是一种二进制格式，可以将特定记录存储在特定于自定义记录的数据类型中。这些数据类型表现为Java类。Sqoop将自动为您生成这些数据类型。此格式支持以二进制表示形式精确存储所有数据，并且适用于存储二进制数据（例如 VARBINARY 列）或将由自定义MapReduce程序主要处理的数据（从SequenceFiles读取比从文本文件读取具有更高的性能），因为不需要解析记录）。

Avro数据文件是一种紧凑，高效的二进制格式，可提供与以其他编程语言编写的应用程序的互操作性。Avro还支持版本控制，因此，例如当在表中添加或删除列时，可以将以前导入的数据文件与新文件一起处理。

默认情况下，不压缩数据。您可以通过将deflate（gzip）算法与 -z 或 --compress 参数一起使用来压缩数据，或者使用 --compression-codec 参数来指定任何Hadoop压缩编解码器 。这适用于SequenceFile，文本和Avro文件。

7.2.12。大对象

Sqoop以特定方式处理大型对象（ BLOB 和 CLOB 列）。如果此数据确实很大，则这些列不应该像大多数列一样在存储器中完全实现。而是以流方式处理其数据。大型对象可以与其余数据内联存储，在这种情况下，每次访问它们都会在内存中完全实现，或者可以将它们存储在链接到主数据存储的辅助存储文件中。默认情况下，大小小于16 MB的大型对象与其余数据一起内联存储。它们较大时会存储在 _lobs 导入目标目录的子目录。这些文件以针对大记录存储进行了优化的单独格式存储，每个文件最多可容纳2 ^ 63字节的记录。 --inline-lob-limit 吊球溢出到单独文件中的大小由参数控制，该 参数采用一个参数，该参数指定要保持内联的最大吊球大小（以字节为单位）。如果将内联LOB限制设置为0，则所有大对象都将放置在外部存储器中。

表6.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符

争论	描述
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段：，行：\n 转义者：\ 可选地，封闭者：'
--optionally-enclosed-by <char>	设置一个字段包围字符

导入定界文件时，定界符的选择很重要。出现在基于字符串的字段中的定界符可能会导致后续分析过程对导入的数据进行模棱两可的解析。例如， "Hello, pleased to meet you" 不应在字段结尾定界符设置为逗号的情况下导入字符串。

分隔符可以指定为：

- 字符（ --fields-terminated-by X ）
- 转义字符（ --fields-terminated-by \t ）。支持的转义字符为：
  - \b （退格键）
  - \n （新队）
  - \r （回车）
  - \t （标签）
  - \" （双引号）
  - \\' （单引号）
  - \\ （反斜杠）
  - \0 （NUL） -这将插入NULL字符字段或线之间，或者将禁用包封/如果用于所述一个逸出 --enclosed-by ， --optionally-enclosed-by 或 --escaped-by 参数。
- UTF-8字符代码点的八进制表示形式。该格式应为 \0ooo ，其中*ooo*是八进制值。例如， --fields-terminated-by \001 将产生 ^A 字符。
- UTF-8字符的代码点的十六进制表示形式。该格式应为 \0xhhh ，其中*hhh*是十六进制值。例如， --fields-terminated-by \0x10 将产生回车符。

默认字段分隔符是逗号（,），记录分隔符是换行符（\n），*无封闭*和*转义*字符。请注意，如果您导入字段数据中包含逗号或换行符的数据库记录，则可能导致记录模糊/无法解析。为了进行明确的解析，必须同时启用两者。例如，通过 --mysql-delimiters 。

如果不能显示明确的定界符，请使用*封闭*和 *转义*字符。（可选）封闭和转义字符的组合将允许对行进行明确的解析。例如，假设数据集的一列包含以下值：

```
Some string, with a comma. Another "string with quotes"
```

以下参数将提供可以明确解析的定界符：

```
$ sqoop import --fields-terminated-by , --escaped-by \\ --enclosed-by '\"' ...
```

（请注意，为防止shell破坏封闭字符，我们已将该参数本身括在单引号中。）

将上述参数应用于上述数据集的结果将是：

```
"Some string, with a comma.","1","2","3"... "Another \"string with quotes\"","4","5","6"...
```

在这里，导入的字符串显示在其他列（ "1","2","3" 等）的上下文中，以演示封装和转义的完整效果。仅当在导入的文本中出现定界符时，才必须使用封闭字符。因此，可以将封闭字符指定为可选字符：

```
$ sqoop import --optionally-enclosed-by '\"' (the rest as above)...
```

这将导致以下导入：

```
"Some string, with a comma.",1,2,3... "Another \"string with quotes\"",4,5,6...
```

笔记 即使Hive支持转义字符，它也不处理换行符的转义。而且，它也不支持可能在封闭字符串中包含字段定界符的封闭字符的概念。因此，在使用Hive时，建议您选择明确的字段和记录终止定界符，而不需要转义和封闭字符。这是由于Hive输入解析能力的限制。

该 --mysql-delimiters 参数是一个速记参数，它使用程序的默认定界符 mysqldump 。如果将 mysqldump 定界符与直接模式导入（带有 --direct ）结合使用，则可以实现非常快的导入。

尽管分隔符的选择对于文本模式导入最重要，但是如果使用导入到SequenceFiles，则分隔符仍然很重要 --as-sequencefile 。生成的类的 toString() 方法将使用您指定的定界符，因此输出数据的后续格式将取决于您选择的定界符。

表7.输入解析参数：

争论	描述
--input-enclosed-by <char>	设置必填字段
--input-escaped-by <char>	设置输入转义字符
--input-fields-terminated-by <char>	设置输入字段分隔符
--input-lines-terminated-by <char>	设置输入的行尾字符
--input-optionally-enclosed-by <char>	设置一个字段包围字符

当Sqoop将数据导入HDFS时，它会生成一个Java类，该类可以重新解释在进行定界格式导入时创建的文本文件。分隔符是使用诸如以下参数来选择的 --fields-terminated-by ： 这既控制如何将数据写入磁盘，又控制生成的 parse() 方法如何重新解释此数据。 parse() 通过使用 --input-fields-terminated-by 等等，可以独立于输出参数选择方法 使用的定界符。例如，这对于生成可以解析使用一组定界符创建的记录并使用单独的定界符集将记录发送到另一组文件的类很有用。

表8. Hive参数：

争论	描述
--hive-home <dir>	覆写 $HIVE_HOME
--hive-import	将表导入到Hive中（如果未设置，则使用Hive的默认定界符。）
--hive-overwrite	覆盖Hive表中的现有数据。

争论	描述
--create-hive-table	如果设置，则目标配置单元将使作业失败
	表存在。默认情况下，此属性为false。
--hive-table <table-name>	设置导入Hive时要使用的表名。
--hive-drop-import-delims	导入到Hive时，从字符串字段中 删除\ *n*，\ *r*和\ *01*。
--hive-delims-replacement	导入到Hive时，将字符串字段中的\ *n*，\ *r*和\ *01* 替换为用户定义的字符串。
--hive-partition-key	要分区的配置单元字段的名称被分片
--hive-partition-value <v>	用作此作业的分区键的字符串值导入到此作业中的蜂巢中。
--map-column-hive <map>	覆盖已配置列的从SQL类型到Hive类型的默认映射。如果在此参数中指定逗号，请使用URL编码的键和值，例如，使用DECIMAL（1%2C%201）而不是DECIMAL（1，1）。

7.2.13。将数据导入到Hive

Sqoop的导入工具的主要功能是将您的数据上传到HDFS中的文件中。如果您具有与HDFS群集关联的Hive元存储，Sqoop还可以通过生成并执行一条 CREATE TABLE 语句来定义Hive中的数据布局，从而将数据导入Hive。将数据导入Hive就像将 **--hive-import** 选项添加到Sqoop命令行一样简单 。

如果Hive表已经存在，则可以指定 **--hive-overwrite** 选项以指示必须替换Hive 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 CREATE TABLE 使用Hive的类型定义您的列的操作，以及一条 LOAD DATA INPATH 将数据文件移至Hive的仓库目录的语句。

该脚本将通过在运行Sqoop的计算机上调用已配置的Hive副本来执行。如果您有多个Hive安装，或者 hive 没有在中 $PATH ，请使用该 **--hive-home** 选项来标识Hive安装目录。Sqoop将 $HIVE_HOME/bin/hive 在此处使用。

笔记 此功能与 --as-avrodatafile 和 不兼容 --as-sequencefile 。

即使Hive支持转义字符，它也不处理换行符的转义。而且，它也不支持可能在封闭字符串中包含字段定界符的封闭字符的概念。因此，在使用Hive时，建议您选择明确的字段和记录终止定界符，而不需要转义和封闭字符。这是由于Hive输入解析能力的限制。如果您确实使用 --escaped-by ， --enclosed-by 或 --optionally-enclosed-by 将数据导入到Hive中，则Sqoop将打印警告消息。

如果数据库的行包含其中包含Hive的默认行定界符（ \n 和 \r 字符）或列定界符（ \01 字符）的字符串字段，则Hive在使用Sqoop导入的数据时会遇到问题。您可以使用该 --hive-drop-import-delims 选项在导入时将那些字符删除以提供与Hive兼容的文本数据。或者，您可以使用该 --hive-delims-replacement 选项在导入时用用户定义的字符串替换那些字符，以提供与Hive兼容的文本数据。仅当您使用Hive的默认定界符时，才应使用这些选项，而如果指定了不同的定界符，则不应使用这些选项。

Sqoop将传递该字段并将记录定界符传递给Hive。如果您未设置任何定界符并使用 --hive-import ，则字段定界符将设置为 ^A ，记录定界符将设置为 \n 与Hive的默认值一致。

默认情况下，Sqoop会将NULL值作为string导入 null 。但是，Hive使用字符串 \N 来表示 NULL 值，因此处理 NULL （如 IS NULL ）的谓词将无法正常工作。你应该追加参数 --null-string 和 --null-non-string 进口工作或情况 --input-null-string ，并 --input-null-non-string 在出口工作的情况下，如果你要妥善保存 NULL 价值。由于sqoop在生成的代码中使用了这些参数，因此您需要正确地将值转义 \N 为 \\N ：

```
$ sqoop import ... --null-string'\\ N'--null-non-string'\\ N'
```

默认情况下，Hive中使用的表名称与源表的名称相同。您可以使用该 --hive-table 选项控制输出表名称。

Hive可以将数据放入分区中，以提高查询性能。您可以通过指定 --hive-partition-key 和 --hive-partition-value 参数，告诉Sqoop作业将Hive的数据导入到特定分区中。分区值必须是字符串。请参阅Hive文档以获取有关分区的更多详细信息。

您可以使用 --compress 和 --compression-codec 选项将压缩表导入Hive 。压缩导入到Hive中的表的缺点之一是，许多并行编解码器无法拆分许多编解码器进行处理。但是，lzop编解码器确实支持拆分。使用此编解码器导入表时，Sqoop将自动为文件建立索引，以使用正确的InputFormat拆分和配置新的Hive表。当前，此功能要求使用lzop编解码器压缩表的所有分区。

表9. HBase参数：

争论	描述
--column-family <family>	设置导入的目标列族
--hbase-create-table	如果指定，请创建缺少的HBase表
--hbase-row-key <col>	指定将哪个输入列用作行键
	如果输入表包含复合
	键，然后<col>必须采用a的形式
	逗号分隔的组合键列表
	属性
--hbase-table <table-name>	指定要用作目标而不是HDFS的HBase表
--hbase-bulkload	启用批量加载

7.2.14。将数据导入HBase

Sqoop支持HDFS和Hive之外的其他导入目标。Sqoop还可以将记录导入到HBase中的表中。

通过指定 --hbase-table ，可以指示Sqoop导入到HBase中的表而不是HDFS中的目录。Sqoop会将数据导入到指定为的参数的表中 --hbase-table 。输入表的每一行将转换为 Put 输出表的一行的HBase 操作。每行的键均取自输入的一列。默认情况下，Sqoop将使用拆分列作为行键列。如果未指定，它将尝试识别源表的主键列（如果有）。您可以使用手动指定行键列 --hbase-row-key 。每个输出列将放在相同的列族中，必须使用来指定 --column-family 。

笔记 此功能与直接导入（参数 --direct ）不兼容 。

如果输入表具有复合键，则 --hbase-row-key 必须采用逗号分隔的复合键属性列表的形式。在这种情况下，将通过使用下划线作为分隔符组合复合键属性的值来生成HBase行的行键。注意：仅 --hbase-row-key 在指定了参数的情况下，具有复合键的表的Sqoop导入才有效。

如果目标表和列族不存在，则Sqoop作业将退出并显示错误。您应该在运行导入之前创建目标表和列系列。如果指定 --hbase-create-table ，则Sqoop将使用HBase配置中的默认参数创建目标表和列系列（如果它们不存在）。

Sqoop当前通过将每个字段转换为其字符串表示形式来将所有值序列化为HBase（就像您在文本模式下导入到HDFS一样），然后将该字符串的UTF-8字节插入目标单元格中。Sqoop将跳过除行键列之外的所有列中所有包含空值的行。

为了减少hbase的负载，Sqoop可以进行批量加载，而不是直接写入。要使用批量加载，请使用启用它 --hbase-bulkload 。

表10. Accumulo参数：

争论	描述
--accumulo-table <table-nam>	指定一个Accumulo表代替HDFS用作目标
--accumulo-column-family <family>	设置导入的目标列族
--accumulo-create-table	如果指定，请创建丢失的Accumulo表
--accumulo-row-key <col>	指定将哪个输入列用作行键
--accumulo-visibility <vis>	（可选）指定可见性令牌，以应用于插入Accumulo的所有行。默认为空字符串。
--accumulo-batch-size <size>	（可选）设置Accumulo的写缓冲区的大小（以字节为单位）。默认值为4MB。
--accumulo-max-latency <ms>	（可选）设置Accumulo批处理写入器的最大延迟（以毫秒为单位）。默认值为0。
--accumulo-zookeepers <host:port>	Accumulo实例使用的逗号分隔的Zookeeper服务器列表
--accumulo-instance <table-name>	目标Accumulo实例的名称
--accumulo-user <username>	导入为的Accumulo用户的名称
--accumulo-password <password>	Accumulo用户的密码

7.2.15。将数据导入Accumulo

Sqoop支持将记录导入Accumulo中的表中

通过指定 --accumulo-table ，可以指示Sqoop导入Accumulo中的表而不是HDFS中的目录。Sqoop会将数据导入到指定为的参数的表中 --accumulo-table 。输入表的每一行将转换为 Mutation 输出表的一行的Accumulo 操作。每行的键均取自输入的一列。默认情况下，Sqoop将使用拆分列作为行键列。如果未指定，它将尝试识别源表的主键列（如果有）。您可以使用手动指定行键列 --accumulo-row-key 。每个输出列将放在相同的列族中，必须使用来指定 --accumulo-column-family 。

> **笔记** 此功能与直接导入（参数 --direct ）不兼容 ，并且不能与HBase导入在同一操作中使用。

如果目标表不存在，则除非 --accumulo-create-table 指定了参数，否则Sqoop作业将退出并显示错误。否则，您应该在运行导入之前创建目标表。

Sqoop当前通过将每个字段转换为其字符串表示形式来将所有值序列化为Accumulo（就像您以文本模式导入到HDFS一样），然后将该字符串的UTF-8字节插入目标单元格中。

默认情况下，不会对Accumulo中的结果单元格应用可见性，因此任何Accumulo用户都可以看到数据。使用 --accumulo-visibility 参数指定可见性令牌，以将其应用于导入作业中的所有行。

要进行性能调整，请使用可选参数 --accumulo-buffer-size\ 和 --accumulo-max-latency 参数。有关这些参数的影响的说明，请参见Accumulo的文档。

为了连接到Accumulo实例，必须使用 --accumulo-zookeepers 参数，Accumulo实例的名称（ --accumulo-instance ）以及用于连接的用户名和密码（ --accumulo-user 和 --accumulo-password ）指定Zookeeper集合的位置。

表11.代码生成参数：

争论	描述
--bindir <dir>	编译对象的输出目录
--class-name <name>	设置生成的类名称。这将覆盖 --package-name 。与结合使用时 --jar-file ，设置输入类别。
--jar-file <file>	禁用代码生成；使用指定的罐子
--outdir <dir>	生成代码的输出目录
--package-name <name>	将自动生成的类放在此包中
--map-column-java <m>	覆盖已配置列的从SQL类型到Java类型的默认映射。

如前所述，将表导入HDFS的副产品是可以处理导入数据的类。如果数据存储在SequenceFiles中，则该类将用于数据的序列化容器。因此，您应该在后续的MapReduce数据处理中使用此类。

该类通常以表命名。名为的表 foo 将生成名为的类 foo 。您可能要覆盖该类名称。例如，如果您的表名为 EMPLOYEES ，则可能需要指定 --class-name Employee 。同样，您可以使用指定包装名称 --package-name 。以下导入将生成一个名为的类 com.foocorp.SomeTable ：

```
$ sqoop import --connect <connect-str> --table SomeTable --package-name com.foocorp
```

.java 您运行的类的源文件将被写入当前的工作目录 sqoop 。您可以使用来控制输出目录 --outdir 。例如， --outdir src/generated/ 。

导入过程将源代码编译为 .class 和 .jar 文件。这些通常存储在 /tmp 。您可以使用选择备用目标目录 --bindir 。例如， --bindir /scratch 。

如果您已经具有可用于执行导入的已编译类，并且希望取消导入过程的代码生成方面，则可以通过提供 --jar-file and --class-name 选项来使用现有的jar和类。例如：

```
$ sqoop import --table SomeTable --jar文件mydatatypes.jar \
    --class-name SomeTableType
```

此命令将从中加载 SomeTableType 类 mydatatypes.jar 。

7.2.16。其他导入配置属性

还有一些其他属性可以通过修改配置 conf/sqoop-site.xml 。可以将属性指定为与Hadoop配置文件中的属性相同，例如：

```
<property>
  <name> property.name </name>
  <value> property.value </value>
</property>
```

也可以在命令行中的通用参数中指定它们，例如：

```
sqoop import -D property.name = property.value ...
```

表12.其他导入配置属性：

争论	描述
sqoop.bigdecimal.format.string	控制以字符串形式存储时BigDecimal列的格式。值 true （默认）将使用toPlainString来存储它们而不包含指数成分（0.0000001）；而值of false 将使用toString，其中可能包含指数（1E-7）
sqoop.hbase.add.row.key	设置为 false （默认）时，Sqoop不会将用作行键的列添加到HBase中的行数据中。设置 true 为时，用作行键的列将添加到HBase中的行数据中。

7.3。示例调用

以下示例说明了在各种情况下如何使用导入工具。

命名表的基本进口 EMPLOYEES 的 corp 数据库：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES
```

需要登录的基本导入：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --username SomeUser -P
Enter password: (hidden)
```

从 EMPLOYEES 表中选择特定列：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --columns "employee_id,first_name,last_n
```

控制导入并行性（使用8个并行任务）：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     -m 8
```

将数据存储在SequenceFiles中，并将生成的类名称设置为 com.foocorp.Employee ：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --class-name com.foocorp.Employee --as-s
```

指定在文本模式导入中使用的定界符：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --fields-terminated-by '\t' --lines-term
```

将数据导入到Hive：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --hive-import
```

仅导入新员工：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --where "start_date > '2010-01-01'"
```

从默认值更改拆分列：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --split-by dept_id
```

验证导入是否成功：

```
$ hadoop fs -ls EMPLOYEES
 Found 5 items
drwxr-xr-x   - someuser somegrp          0 2010-04-27 16:40 /user/someuser/EMPLOYEES/_logs
-rw-r--r--   1 someuser somegrp    2913511 2010-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00000
-rw-r--r--   1 someuser somegrp    1683938 2010-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00001
-rw-r--r--   1 someuser somegrp    7245839 2010-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00002
-rw-r--r--   1 someuser somegrp    7842523 2010-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00003

$ hadoop fs -cat EMPLOYEES/part-m-00000 | head -n 10 0,joe,smith,engineering 1,jane,doe,marketing
```

在已经导入表的前100,000行之后，执行新数据的增量导入：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/somedb --table sometable \     --where "id > 100000" --target-dir /in
```

EMPLOYEES 在 corp 数据库中命名的表的导入，该表使用验证通过表行数和复制到HDFS的行数来验证导入： 更多详细信息

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp \
--table EMPLOYEES --validate
```

8。 sqoop-import-all-tables

8.1。目的

该 import-all-tables 工具将一组表从RDBMS导入到HDFS。每个表中的数据都存储在HDFS的单独目录中。

为了使该 import-all-tables 工具有用，必须满足以下条件：

- 每个表必须具有单列主键，或者 --autoreset-to-one-mapper 必须使用选项。
- 您必须打算导入每个表的所有列。
- You must not intend to use non-default splitting column, nor impose any conditions via a WHERE clause.

8.2. Syntax

```
$ sqoop import-all-tables (generic-args) (import-args)
$ sqoop-import-all-tables (generic-args) (import-args)
```

Although the Hadoop generic arguments must preceed any import arguments, the import arguments can be entered in any order with respect to one another.

Table 13. Common arguments

Argument	Description
--connect <jdbc-uri>	Specify JDBC connect string
--connection-manager <class-name>	Specify connection manager class to use
--driver <class-name>	Manually specify JDBC driver class to use
--hadoop-mapred-home <dir>	Override $HADOOP_MAPRED_HOME
--help	Print usage instructions
--password-file	Set path for a file containing the authentication password
-P	Read password from console
--password <password>	Set authentication password
--username <username>	Set authentication username
--verbose	Print more information while working
--connection-param-file <filename>	Optional properties file that provides connection parameters
--relaxed-isolation	Set connection transaction isolation to read uncommitted for the mappers.

Table 14. Import control arguments:

Argument	Description
--as-avrodatafile	Imports data to Avro Data Files
--as-sequencefile	Imports data to SequenceFiles
--as-textfile	Imports data as plain text (default)
--as-parquetfile	Imports data to Parquet Files
--direct	Use direct import fast path
--inline-lob-limit <n>	Set the maximum size for an inline LOB
-m,--num-mappers <n>	Use *n* map tasks to import in parallel
--warehouse-dir <dir>	HDFS parent for table destination
-z,--compress	启用压缩
--compression-codec <c>	使用Hadoop编解码器（默认gzip）
--exclude-tables <tables>	用逗号分隔的表列表，以排除在导入过程之外
--autoreset-to-one-mapper	如果遇到没有主键的表，则导入应使用一个映射器

这些参数的行为作为用于当他们这样做同样的方式 sqoop-import 的工具，但是 --table ， --split-by ， --columns ，和 --where 参数是无效的 sqoop-import-all-tables 。该 --exclude-tables argument is for +sqoop-import-all-tables 只。

表15.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段： ，行： \n 转义者： \ 可选地，封闭者： '

争论	描述
--optionally-enclosed-by <char>	设置一个字段包围字符

表16.输入解析参数：

争论	描述
--input-enclosed-by <char>	设置必填字段
--input-escaped-by <char>	设置输入转义字符
--input-fields-terminated-by <char>	设置输入字段分隔符
--input-lines-terminated-by <char>	设置输入的行尾字符
--input-optionally-enclosed-by <char>	设置一个字段包围字符

表17. Hive参数：

争论	描述
--hive-home <dir>	覆写 $HIVE_HOME
--hive-import	将表导入到Hive中（如果未设置，则使用Hive的默认定界符。）
--hive-overwrite	覆盖Hive表中的现有数据。
--create-hive-table	如果设置，则目标配置单元将使作业失败
	表存在。默认情况下，此属性为false。
--hive-table <table-name>	设置导入Hive时要使用的表名。
--hive-drop-import-delims	导入到Hive时，从字符串字段中 删除\ *n*，\ *r*和\ *01*。
--hive-delims-replacement	导入到Hive时，将字符串字段中的\ *n*，\ *r*和\ *01* 替换为用户定义的字符串。

争论	描述
--hive-partition-key	要分区的配置单元字段的名称被分片
--hive-partition-value <v>	用作此作业的分区键的字符串值导入到此作业中的蜂巢中。
--map-column-hive <map>	覆盖已配置列的从SQL类型到Hive类型的默认映射。如果在此参数中指定逗号，请使用URL编码的键和值，例如，使用DECIMAL（1%2C%201）而不是DECIMAL（1，1）。

表18.代码生成参数：

争论	描述
--bindir <dir>	编译对象的输出目录
--jar-file <file>	禁用代码生成；使用指定的罐子
--outdir <dir>	生成代码的输出目录
--package-name <name>	将自动生成的类放在此包中

该 import-all-tables 工具不支持该 --class-name 参数。但是，您可以指定一个包，将 --package-name 在其中放置所有生成的类。

8.3。示例调用

从 corp 数据库导入所有表：

```
$ sqoop import-all-tables --connect jdbc：mysql：//db.foo.com/corp
```

验证它是否有效：

```
$ hadoop fs -ls
找到4项
drwxr-xr-x-someuser somegrp 0 2010-04-27 17:15 / user / someuser / EMPLOYEES
drwxr-xr-x-someuser somegrp 0 2010-04-27 17:15 / user / someuser / PAYCHECKS
drwxr-xr-x-someuser somegrp 0 2010-04-27 17:15 / user / someuser /部门
drwxr-xr-x-someuser somegrp 0 2010-04-27 17:15 / user / someuser / OFFICE_SUPPLIES
```

9。 sqoop-import-mainframe

9.1。目的

该 import-mainframe 工具将大型机上的分区数据集（PDS）中的所有顺序数据集导入到HDFS。PDS类似于开放系统上的目录。数据集中的记录只能包含字符数据。记录将与整个记录一起存储为单个文本字段。

9.2。句法

$ sqoop import-mainframe（generic-args） （import-args） $ sqoop-import-mainframe（generic-args） （import-args）

尽管Hadoop通用参数必须在任何导入参数之前，但是您可以相对于彼此以任何顺序键入导入参数。

表19.常用参数

争论	描述
--connect <hostname>	指定要连接的主机
--connection-manager <class-name>	指定要使用的连接管理器类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME

争论	描述
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件

9.2.1。连接到大型机

Sqoop旨在将大型机数据集导入HDFS。为此，必须在Sqoop --connect 参数中指定大型机主机名。

```
$ sqoop import-mainframe --connect z390
```

这将通过ftp连接到大型机主机z390。

您可能需要针对大型主机进行身份验证才能访问它。您可以使用将 --username 用户名提供给大型机。Sqoop提供了几种不同的方式来向大型机提供安全和非安全的密码，下面将对此进行详细介绍。

向主机提供密码的安全方式。 您应该将密码保存在用户主目录中的文件中，该文件具有400个权限，并使用 **--password-file** 参数指定该文件的路径，这是输入凭据的首选方法。然后，Sqoop将使用安全方式从文件中读取密码，并将其传递给MapReduce集群，而不会在作业配置中公开密码。包含密码的文件可以在本地FS或HDFS上。

例子：

```
$ sqoop import-mainframe --connect z390 \
    --username大卫--password文件$ {user.home} /。password
```

提供密码的另一种方法是使用 -P 参数，该参数将从控制台提示中读取密码。

警告 该 --password 参数不安全，因为其他用户可能会通过诸如的程序输出从命令行参数读取您的密码 ps 。与使用 **-P** 参数相比，参数是首选方法 --password 。凭证仍可以使用不安全手段在MapReduce群集的节点之间传输。

例子：

```
$ sqoop import-mainframe --connect z390 --username david --password 12345
```

表20.导入控制参数：

争论	描述
--as-avrodatafile	将数据导入Avro数据文件
--as-sequencefile	将数据导入到SequenceFiles
--as-textfile	以纯文本格式导入数据（默认）
--as-parquetfile	将数据导入Parquet文件
--delete-target-dir	删除导入目标目录（如果存在）
-m,--num-mappers <n>	使用*n*个地图任务并行导入
--target-dir <dir>	HDFS目标目录
--warehouse-dir <dir>	表目标的HDFS父级
-z,--compress	启用压缩
--compression-codec <c>	使用Hadoop编解码器（默认gzip）

9.2.2。选择要导入的文件

您可以使用该 --dataset 参数指定分区的数据集名称。分区数据集中的所有顺序数据集都将被导入。

9.2.3。控制并行

Sqoop通过与大型机建立多个ftp连接来并行导入数据，以同时传输多个文件。您可以使用 -m 或 --num-mappers 参数指定用于执行导入的映射任务（并行进程）的数量。这些参数中的每个参数取一个整数值，该整数值与要采用的并行度相对应。默认情况下，使用四个任务。您可以调整该值，以最大程度地提高大型机的数据传输速率。

9.2.4。控制分布式缓存

每次启动Sqoop作业时，Sqoop都会将$ SQOOP_HOME / lib文件夹中的jar复制到作业缓存中。由Oozie启动时，这是不必要的，因为Oozie使用自己的Sqoop共享库，该库将Sqoop依赖项保留在分布式缓存中。Oozie将在第一个Sqoop作业期间仅在每个工作节点上对Sqoop依赖项进行本地化，然后将工作节点上的jar重新用于次要工作。 --skip-dist-cache 由Oozie启动时，在Sqoop命令中使用option将跳过Sqoop将其依赖项复制到作业缓存并节省大量I / O的步骤。

9.2.5。控制导入过程

默认情况下，Sqoop会将分区数据集中的所有顺序文件导入 pds 到 pds HDFS主目录内的目录中。例如，如果您的用户名是 someuser ，则导入工具将写入 /user/someuser/pds/(files) 。您可以使用 --warehouse-dir 参数调整导入的父目录。例如：

```
$ sqoop import-mainframe --connnect <host> --dataset foo --warehouse-dir / shared \
    ...
```

该命令将写入目录中的一组文件 /shared/pds/ 。

您还可以显式选择目标目录，如下所示：

```
$ sqoop import-mainframe --connnect <主机> --dataset foo --target-dir / dest \
    ...
```

这会将文件导入 /dest 目录。 --target-dir 与不兼容 --warehouse-dir 。

默认情况下，导入将转到新的目标位置。如果目标目录已存在于HDFS中，则Sqoop将拒绝导入并覆盖该目录的内容。

9.2.6。档案格式

默认情况下，数据集中的每个记录都存储为文本记录，结尾处带有换行符。假定每个记录包含一个名称为DEFAULT_COLUMN的单个文本字段。当Sqoop将数据导入HDFS时，它将生成一个Java类，该类可以重新解释其创建的文本文件。

您也可以将大型机记录导入到Sequence，Avro或Parquet文件中。

默认情况下，不压缩数据。您可以通过将deflate（gzip）算法与 -z 或 --compress 参数一起使用来压缩数据，或者使用 --compression-codec 参数来指定任何Hadoop压缩编解码器 。

表21.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段： ，行： \n 转义者： \ 可选地，封闭者： '

争论	描述
--optionally-enclosed-by <char>	设置一个字段包围字符

由于大型机记录仅包含一个字段，因此导入到带分隔符的文件将不包含任何字段分隔符。但是，该字段可以用括起来的字符括起来，也可以用转义字符将其转义。

表22.输入解析参数：

争论	描述
--input-enclosed-by <char>	设置必填字段
--input-escaped-by <char>	设置输入转义字符
--input-fields-terminated-by <char>	设置输入字段分隔符
--input-lines-terminated-by <char>	设置输入的行尾字符
--input-optionally-enclosed-by <char>	设置一个字段包围字符

当Sqoop将数据导入HDFS时，它会生成一个Java类，该类可以重新解释在进行定界格式导入时创建的文本文件。分隔符是使用诸如以下参数来选择的 --fields-terminated-by ： 这既控制如何将数据写入磁盘，又控制生成的 parse() 方法如何重新解释此数据。 parse() 通过使用 --input-fields-terminated-by 等等，可以独立于输出参数选择方法 使用的定界符。例如，这对于生成可以解析使用一组定界符创建的记录并使用单独的定界符集将记录发送到另一组文件的类很有用。

表23. Hive参数：

争论	描述
--hive-home <dir>	覆写 $HIVE_HOME

争论	描述
--hive-import	将表导入到Hive中（如果未设置，则使用Hive的默认定界符。）
--hive-overwrite	覆盖Hive表中的现有数据。
--create-hive-table	如果设置，则目标配置单元将使作业失败
	表存在。默认情况下，此属性为false。
--hive-table <table-name>	设置导入Hive时要使用的表名。
--hive-drop-import-delims	导入到Hive时，从字符串字段中 删除\ *n*，\ *r*和\ *01*。
--hive-delims-replacement	导入到Hive时，将字符串字段中的\ *n*，\ *r*和\ *01* 替换为用户定义的字符串。
--hive-partition-key	要分区的配置单元字段的名称被分片
--hive-partition-value <v>	用作此作业的分区键的字符串值导入到此作业中的蜂巢中。
--map-column-hive <map>	覆盖已配置列的从SQL类型到Hive类型的默认映射。如果在此参数中指定逗号，请使用URL编码的键和值，例如，使用DECIMAL（1%2C%201）而不是DECIMAL（1，1）。

9.2.7。将数据导入到Hive

Sqoop的导入工具的主要功能是将您的数据上传到HDFS中的文件中。如果您具有与HDFS群集关联的Hive元存储，Sqoop还可以通过生成并执行一条 CREATE TABLE 语句来定义Hive中的数据布局，从而将数据导入Hive。将数据导入Hive就像将 **--hive-import** 选项添加到Sqoop命令行一样简单 。

如果Hive表已经存在，则可以指定 **--hive-overwrite** 选项以指示必须替换Hive 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 CREATE TABLE 使用Hive的类型定义您的列的操作，以及一条 LOAD DATA INPATH 将数据文件移至Hive的仓库目录的语句。

该脚本将通过在运行Sqoop的计算机上调用已配置的Hive副本来执行。如果您有多个Hive安装，或者 hive 没有在中 $PATH ，请使用该 **--hive-home** 选项来标识Hive安装目录。Sqoop将 $HIVE_HOME/bin/hive 在此处使用。

> **笔记** 此功能与 --as-avrodatafile 和 不兼容 --as-sequencefile 。

即使Hive支持转义字符，它也不处理换行符的转义。而且，它也不支持可能在封闭字符串中包含字段定界符的封闭字符的概念。因此，在使用Hive时，建议您选择明确的字段和记录终止定界符，而不需要转义和封闭字符。这是由于Hive输入解析能力的限制。如果您确实使用 --escaped-by ， --enclosed-by 或 --optionally-enclosed-by 将数据导入到Hive中，则Sqoop将打印警告消息。

如果数据库的行包含其中包含Hive的默认行定界符（ \n 和 \r 字符）或列定界符（ \01 字符）的字符串字段，则Hive在使用Sqoop导入的数据时会遇到问题。您可以使用该 --hive-drop-import-delims 选项在导入时将那些字符删除以提供与Hive兼容的文本数据。或者，您可以使用该 --hive-delims-replacement 选项在导入时用用户定义的字符串替换那些字符，以提供与Hive兼容的文本数据。仅当您使用Hive的默认定界符时，才应使用这些选项，而如果指定了不同的定界符，则不应使用这些选项。

Sqoop将传递该字段并将记录定界符传递给Hive。如果您未设置任何定界符并使用 --hive-import ，则字段定界符将设置为 ^A ，记录定界符将设置为 \n 与Hive的默认值一致。

默认情况下，Sqoop会将NULL值作为string导入 null 。但是，Hive使用字符串 \N 来表示 NULL 值，因此处理 NULL （如 IS NULL ）的谓词将无法正常工作。你应该追加参数 --null-string 和 --null-non-string 进口工作或情况 --input-null-string ，并 --input-null-non-string 在出口工作的情况下，如果你要妥善保存 NULL 价值。由于sqoop在生成的代码中使用了这些参数，因此您需要正确地将值转义 \N 为 \\N ：

```
$ sqoop import ... --null-string'\\ N'--null-non-string'\\ N'
```

默认情况下，Hive中使用的表名称与源表的名称相同。您可以使用该 --hive-table 选项控制输出表名称。

Hive可以将数据放入分区中，以提高查询性能。您可以通过指定 --hive-partition-key 和 --hive-partition-value 参数，告诉Sqoop作业将Hive的数据导入到特定分区中。分区值必须是字符串。请参阅Hive文档以获取有关分区的更多详细信息。

您可以使用 --compress 和 --compression-codec 选项将压缩表导入Hive 。压缩导入到Hive中的表的缺点之一是，许多并行编解码器无法拆分许多编解码器进行处理。但是，lzop编解码器确实支持拆分。使用此编解码器导入表时，Sqoop将自动为文件建立索引，以使用正确的InputFormat拆分和配置新的Hive表。当前，此功能要求使用lzop编解码器压缩表的所有分区。

表24. HBase参数：

争论	描述
--column-family <family>	设置导入的目标列族
--hbase-create-table	如果指定，请创建缺少的HBase表
--hbase-row-key <col>	指定将哪个输入列用作行键
	如果输入表包含复合
	键，然后<col>必须采用a的形式
	逗号分隔的组合键列表
	属性
--hbase-table <table-name>	指定要用作目标而不是HDFS的HBase表
--hbase-bulkload	启用批量加载

9.2.8。将数据导入HBase

Sqoop支持HDFS和Hive之外的其他导入目标。Sqoop还可以将记录导入到HBase中的表中。

通过指定 --hbase-table ，可以指示Sqoop导入到HBase中的表而不是HDFS中的目录。Sqoop会将数据导入到指定为的参数的表中 --hbase-table 。输入表的每一行将转换为 Put 输出表的一行的HBase 操作。每行的键均取自输入的一列。默认情况下，Sqoop将使用拆分列作为行键列。如果未指定，它将尝试识别源表的主键列（如果有）。您可以使用手动指定行键列 --hbase-row-key 。每个输出列将放在相同的列族中，必须使用来指定 --column-family 。

笔记 此功能与直接导入（参数 --direct ）不兼容 。

如果输入表具有复合键，则 --hbase-row-key 必须采用逗号分隔的复合键属性列表的形式。在这种情况下，将通过使用下划线作为分隔符组合复合键属性的值来生成HBase行的行键。注意：仅 --hbase-row-key 在指定了参数的情况下，具有复合键的表的Sqoop导入才有效。

如果目标表和列族不存在，则Sqoop作业将退出并显示错误。您应该在运行导入之前创建目标表和列系列。如果指定 --hbase-create-table ，则Sqoop将使用HBase配置中的默认参数创建目标表和列系列（如果它们不存在）。

Sqoop当前通过将每个字段转换为其字符串表示形式来将所有值序列化为HBase（就像您在文本模式下导入到HDFS一样），然后将该字符串的UTF-8字节插入目标单元格中。Sqoop将跳过除行键列之外的所有列中所有包含空值的行。

为了减少hbase的负载，Sqoop可以进行批量加载，而不是直接写入。要使用批量加载，请使用启用它 --hbase-bulkload 。

表25. Accumulo参数：

争论	描述
--accumulo-table <table-nam>	指定一个Accumulo表代替HDFS用作目标
--accumulo-column-family <family>	设置导入的目标列族
--accumulo-create-table	如果指定，请创建丢失的Accumulo表
--accumulo-row-key <col>	指定将哪个输入列用作行键
--accumulo-visibility <vis>	（可选）指定可见性令牌，以应用于插入Accumulo的所有行。默认为空字符串。
--accumulo-batch-size <size>	（可选）设置Accumulo的写缓冲区的大小（以字节为单位）。默认值为4MB。
--accumulo-max-latency <ms>	（可选）设置Accumulo批处理写入器的最大延迟（以毫秒为单位）。默认值为0。
--accumulo-zookeepers <host:port>	Accumulo实例使用的逗号分隔的Zookeeper服务器列表
--accumulo-instance <table-name>	目标Accumulo实例的名称

争论	描述
--accumulo-user <username>	导入为的Accumulo用户的名称
--accumulo-password <password>	Accumulo用户的密码

9.2.9。将数据导入Accumulo

Sqoop支持将记录导入Accumulo中的表中

通过指定 --accumulo-table ，可以指示Sqoop导入Accumulo中的表而不是HDFS中的目录。Sqoop会将数据导入到指定为的参数的表中 --accumulo-table 。输入表的每一行将转换为 Mutation 输出表的一行的Accumulo 操作。每行的键均取自输入的一列。默认情况下，Sqoop将使用拆分列作为行键列。如果未指定，它将尝试识别源表的主键列（如果有）。您可以使用手动指定行键列 --accumulo-row-key 。每个输出列将放在相同的列族中，必须使用来指定 --accumulo-column-family 。

笔记 此功能与直接导入（参数 --direct ）不兼容 ，并且不能与HBase导入在同一操作中使用。

如果目标表不存在，则除非 --accumulo-create-table 指定了参数，否则Sqoop作业将退出并显示错误。否则，您应该在运行导入之前创建目标表。

Sqoop当前通过将每个字段转换为其字符串表示形式来将所有值序列化为Accumulo（就像您以文本模式导入到HDFS一样），然后将该字符串的UTF-8字节插入目标单元格中。

默认情况下，不会对Accumulo中的结果单元格应用可见性，因此任何Accumulo用户都可以看到数据。使用 --accumulo-visibility 参数指定可见性令牌，以将其应用于导入作业中的所有行。

要进行性能调整，请使用可选参数 --accumulo-buffer-size\ 和 --accumulo-max-latency 参数。有关这些参数的影响的说明，请参见Accumulo的文档。

为了连接到Accumulo实例，必须使用 --accumulo-zookeepers 参数，Accumulo实例的名称（ --accumulo-instance ）以及用于连接的用户名和密码（ --accumulo-user 和 --accumulo-password ）指定Zookeeper集合的位置。

表26.代码生成参数：

争论	描述
--bindir <dir>	编译对象的输出目录
--class-name <name>	设置生成的类名称。这将覆盖 --package-name 。与结合使用时 --jar-file ，设置输入类别。
--jar-file <file>	禁用代码生成；使用指定的罐子
--outdir <dir>	生成代码的输出目录
--package-name <name>	将自动生成的类放在此包中
--map-column-java <m>	覆盖已配置列的从SQL类型到Java类型的默认映射。

如前所述，将表导入HDFS的副产品是可以处理导入数据的类。您应该在后续的MapReduce数据处理中使用此类。

该类通常以分区的数据集名称命名。名为的分区数据集 foo 将生成一个名为的类 foo 。您可能要覆盖该类名称。例如，如果您的分区数据集名为 EMPLOYEES ，则可能需要指定 --class-name Employee 。同样，您可以使用指定包装名称 --package-name 。以下导入将生成一个名为的类 com.foocorp.SomePDS ：

```
$ sqoop import-mainframe --connect <host> --dataset SomePDS --package-name com.foocorp
```

.java 您运行的类的源文件将被写入当前的工作目录 sqoop 。您可以使用来控制输出目录 --outdir 。例如， --outdir src/generated/ 。

导入过程将源代码编译为 .class 和 .jar 文件。这些通常存储在 /tmp 。您可以使用选择备用目标目录 --bindir 。例如， --bindir /scratch 。

如果您已经具有可用于执行导入的已编译类，并且希望取消导入过程的代码生成方面，则可以通过提供 --jar-file and --class-name 选项来使用现有的jar和类。例如：

```
$ sqoop import-mainframe --dataset SomePDS --jar文件mydatatypes.jar \
    --class-name SomePDSType
```

此命令将从中加载 SomePDSType 类 mydatatypes.jar 。

9.2.10。其他导入配置属性

还有一些其他属性可以通过修改配置 conf/sqoop-site.xml 。可以将属性指定为与Hadoop配置文件中的属性相同，例如：

```
  <属性>
    <name> property.name </ name>
    <value> property.value </ value>
  </ property>
```

也可以在命令行中的通用参数中指定它们，例如：

```
sqoop import -D property.name = property.value ...
```

9.3。示例调用

以下示例说明了在各种情况下如何使用导入工具。

基本导入 EMPLOYEES 主机主机z390中命名的分区数据集中的所有顺序文件：

```
$ sqoop import-mainframe --connect z390 --dataset EMPLOYEES \
--username SomeUser -P Enter password: (hidden)
```

控制导入并行性（使用8个并行任务）：

```
$ sqoop import-mainframe --connect z390 --dataset EMPLOYEES \
--username SomeUser --password-file mypassword -m 8
```

将数据导入到Hive：

```
$ sqoop import-mainframe --connect z390 --dataset EMPLOYEES \
--hive-import
```

10。 sqoop-export

10.1。目的

该 export 工具将一组文件从HDFS导出回RDBMS。目标表必须已经存在于数据库中。根据用户指定的定界符，读取输入文件并将其解析为一组记录。

缺省操作是将它们转换为一组 INSERT 将记录注入数据库的语句。在“更新模式”下，Sqoop将生成 UPDATE 替换数据库中现有记录的语句，在“调用模式”下，Sqoop将为每个记录进行存储过程调用。

10.2。句法

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

尽管Hadoop通用参数必须在任何导出参数之前，但是可以相对于彼此以任何顺序输入导出参数。

表27.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

表28.验证参数更多详细信息

争论	描述
--validate	启用对复制数据的验证，仅支持单表复制。
--validator <class-name>	指定要使用的验证器类。
--validation-threshold <class-name>	指定要使用的验证阈值类。
--validation-failurehandler <class-name>	指定要使用的验证失败处理程序类。

表29.导出控制参数：

参数	描述
--columns <col,col,col…>	要导出到表的列
--direct	使用直接导出快速路径
--export-dir <dir>	HDFS导出的源路径
-m,--num-mappers <n>	使用*n*个地图任务并行导出
--table <table-name>	要填充的表
--call <stored-proc-name>	存储过程调用
--update-key <col-name>	用于更新的锚列。如果有多个列，请使用逗号分隔的列列表。
--update-mode <mode>	指定在数据库中使用不匹配的键找到新行时如何执行更新。
	mode 包括 updateonly （默认）和的 合法值 allowinsert 。
--input-null-string <null-string>	字符串列将被解释为null的字符串
--input-null-non-string <null-string>	非字符串列将被解释为null的字符串
--staging-table <staging-table-name>	在将数据插入目标表之前将在其中暂存的表。
--clear-staging-table	指示可以删除暂存表中存在的任何数据。
--batch	使用批处理模式执行基础语句。

该 --export-dir 参数和一个 --table 或者 --call 是必需的。它们指定要在数据库中填充的表（或要调用的存储过程），以及HDFS中包含源数据的目录。

默认情况下，将选择表中的所有列进行导出。您可以选择一个列子集，并通过使用 --columns 参数来控制列的顺序 。这应该包括以逗号分隔的要导出列的列表。例如： --columns "col1,col2,col3" 。请注意，未包含在 --columns 参数中的列需要具有定义的默认值或允许 NULL 值。否则，您的数据库将拒绝导入的数据，从而使Sqoop作业失败。

您可以独立于目录中存在的文件数量来控制映射器的数量。导出性能取决于并行度。**默认情况下，Sqoop将在导出过程中并行使用四个任务**。这可能不是最佳选择。您将需要尝试使用自己的特定设置。其他任务可能会提供更好的并发性，但是如果数据库已经在更新索引，调用触发器等方面成为瓶颈，那么额外的负载可能会降低性能。的 --num-mappers 或 -m 参数控制的地图任务的数量，这是并行的使用程度。

一些数据库也为导出提供了直接模式。使用 --direct 参数指定此代码路径。这可能比标准JDBC代码路径具有更高的性能。有关在每个特定RDBMS上使用直接模式的详细信息，安装要求，可用选项和限制，请参见第25节“特定连接器的注意事项”。

在 --input-null-string 和 --input-null-non-string 参数都是可选的。如果 --input-null-string 未指定，则字符串类型列的字符串“ null”将被解释为null。如果 --input-null-non-string 未指定，则对于非字符串列，字符串“ null”和空字符串都将被解释为null。请注意，除了由所指定的其他字符串外，对于非字符串列，空字符串将始终被解释为null --input-null-non-string 。

由于Sqoop将导出过程分解为多个事务，因此失败的导出作业可能会导致部分数据提交到数据库。在某些情况下，这可能进一步导致后续作业因插入冲突而失败，而在其他情况下，则可能导致数据重复。您可以通过该 --staging-table 选项指定一个临时表来克服此问题，该选项用作用于暂存导出数据的辅助表。最后，已分阶段处理的数据将在单个事务中移至目标表。

为了使用登台工具，必须在运行导出作业之前创建登台表。该表在结构上必须与目标表相同。该表在导出作业运行之前应该为空，或者 --clear-staging-table 必须指定该选项。如果登台表包含数据并且 --clear-staging-table 指定了选项，则Sqoop将在开始导出作业之前删除所有数据。

> **笔记** 对于将暂存数据推送到目标表之前的暂存数据的支持并不总是可用于 --direct 导出。当使用 --update-key 用于更新现有数据的选项调用导出时，以及使用存储过程插入数据时，该功能也不可用。最好检查第25节“特定连接器的注意事项”部分进行验证。

10.3。插入与更新

默认情况下， sqoop-export 将新行追加到表；每个输入记录都转换为一条 INSERT 语句，该语句在目标数据库表中添加了一行。如果表具有约束（例如，其键值必须唯一的主键列）并且已经包含数据，则必须注意避免插入违反这些约束的记录。如果 INSERT 语句失败，导出过程将失败。此模式主要用于将记录导出到旨在接收这些结果的新的空表中。

如果指定该 --update-key 参数，则Sqoop将改为修改数据库中的现有数据集。每个输入记录都被视为 UPDATE 修改现有行的语句。语句修改的行由用指定的列名确定 --update-key 。例如，考虑以下表定义：

```
CREATE TABLE foo(
     id INT NOT NULL PRIMARY KEY,
     msg VARCHAR(32),
     bar INT);
```

还考虑HDFS中包含如下记录的数据集：

```
0,this is a test,42
1,some more data,100
 ...
```

运行 sqoop-export --table foo --update-key id --export-dir /path/to/data --connect … 将运行一个导出作业，该导出作业将基于以下数据执行SQL语句：

```
UPDATE foo SET msg='this is a test', bar=42 WHERE id=0;
UPDATE foo SET msg='some more data', bar=100 WHERE id=1;
 ...
```

如果一条 UPDATE 语句不修改任何行，则不认为这是错误；出口将无声地继续。（实际上，这意味着基于更新的导出将不会在数据库中插入新行。）同样，如果用指定的列 --update-key 不能唯一地标识行，并且用单个语句更新了多行，则也不会检测到此情况。

--update-key 也可以给该参数一个逗号分隔的列名列表。在这种情况下，Sqoop将在更新任何现有记录之前匹配此列表中的所有键。

根据目标数据库的不同，如果要更新数据库中已经存在的行，或者如果还不存在则插入行，则还可以 --update-mode 使用 allowinsert mode指定参数。

表30.输入解析参数：

争论	描述
--input-enclosed-by <char>	设置必填字段
--input-escaped-by <char>	设置输入转义字符
--input-fields-terminated-by <char>	设置输入字段分隔符
--input-lines-terminated-by <char>	设置输入的行尾字符
--input-optionally-enclosed-by <char>	设置一个字段包围字符

表31.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段：，行：\n 转义者：\ 可选地，封闭者：'
--optionally-enclosed-by <char>	设置一个字段包围字符

Sqoop自动生成代码以解析和解释包含要导出回数据库的数据的文件记录。如果这些文件是使用非默认定界符（用逗号分隔的字段和换行符分隔的记录）创建的，则应再次指定相同的定界符，以便Sqoop可以解析您的文件。

如果您指定了不正确的分隔符，则Sqoop将无法在每行中找到足够的列。这将导致输出映射任务因throw而失败 ParseExceptions 。

表32.代码生成参数：

争论	描述
--bindir <dir>	编译对象的输出目录
--class-name <name>	设置生成的类名称。这将覆盖 --package-name 。与结合使用时 --jar-file ，设置输入类别。
--jar-file <file>	禁用代码生成；使用指定的jar文件
--outdir <dir>	生成代码的输出目录
--package-name <name>	将自动生成的类放在此包中
--map-column-java <m>	覆盖已配置列的从SQL类型到Java类型的默认映射。

如果要导出的记录是先前导入的结果生成的，则可以使用原始生成的类将数据读回。在这种情况下，指定 --jar-file 并 --class-name 消除了指定分隔符的需要。

现有生成代码的使用与 --update-key ；更新模式导出需要新的代码生成来执行更新。您不能使用 --jar-file ， 并且必须完全指定任何非默认的定界符。

10.4。导出和事务

> 导出由多个 <code>writers</code>并行执行。每个作者使用一个单独的数据库连接；这些具有彼此独立的事务。Sqoop使用多行<code>INSERT</code>语

10.5。导出失败

出口可能由于多种原因而失败：

- Hadoop集群与数据库之间的连接丢失（由于硬件故障或服务器软件崩溃）
- 尝试 INSERT 违反一致性约束的行（例如，插入重复的主键值）
- 尝试从HDFS源数据中解析不完整或格式不正确的记录
- 尝试使用不正确的定界符解析记录
- 容量问题（例如RAM或磁盘空间不足）

如果导出映射任务由于这些或其他原因而失败，则将导致导出作业失败。导出失败的结果是不确定的。每个导出映射任务都在单独的事务中运行。此外，个别地图会 commit 定期执行当前交易任务。如果任务失败，则当前事务将回滚。任何先前提交的事务将在数据库中保持持久性，从而导致部分完成导出。

10.6。示例调用

基本导出，用于填充名为的表 bar ：

```
$ sqoop export --connect jdbc:mysql://db.example.com/foo --table bar  \     --export-dir /results/bar_data
```

本示例将文件放入， /results/bar_data 并将其内容注入 bar 上 foo 数据库的表中 db.example.com 。目标表必须已经存在于数据库中。Sqoop执行一组 INSERT INTO 操作，而不考虑现有内容。如果Sqoop尝试插入违反数据库约束的行（例如，已经存在特定的主键值），则导出将失败。

另外，您可以通过提供指定要导出的列 --columns "col1,col2,col3" 。请注意，未包含在 --columns 参数中的列必须具有定义的默认值或允许 NULL 值。否则，您的数据库将拒绝导入的数据，从而使Sqoop作业失败。

另一个用于填充 bar 启用了验证功能的表的基本导出： 更多详细信息

```
$ sqoop export --connect jdbc:mysql://db.example.com/foo --table bar  \     --export-dir /results/bar_data --validate
```

调用 barproc 为每个记录 命名的存储过程的导出 /results/bar_data 看起来像：

```
$ sqoop export --connect jdbc:mysql://db.example.com/foo --call barproc \     --export-dir /results/bar_data
```

11。 validation

11.1。目的

通过比较源副本和目标副本中的行数来验证复制的数据（导入还是导出）。

11.2。介绍

有3个基本接口：ValidationThreshold-确定源和目标之间的误差范围是否可以接受：绝对值，百分比容限等。默认实现是AbsoluteValidationThreshold，可确保源和目标中的行数相同。

ValidationFailureHandler-负责处理故障：记录错误/警告，中止等。默认实现是LogOnFailureHandler，它将警告消息记录到已配置的记录器中。

验证程序-通过将决策委派给ValidationThreshold并将失败处理委派给ValidationFailureHandler来驱动验证逻辑。默认实现是RowCountValidator，它可以验证源和目标中的行数。

11.3。句法

```
$ sqoop import（generic-args）（import-args）
$ sqoop export（泛型参数）（export-args）
```

验证参数是导入和导出参数的一部分。

11.4。Configuration

验证框架是可扩展的和可插入的。它带有默认实现，但是可以通过将接口作为命令行参数的一部分传递来扩展接口以允许自定义实现，如下所述。

Validator。

```
Property:          validator
Description:       Driver for validation,
                   must implement org.apache.sqoop.validation.Validator Supported values: The value has to be a fully qu
Default value:     org.apache.sqoop.validation.RowCountValidator
```

Validation Threshold 。

```
Property:          validation-threshold
Description:       Drives the decision based on the validation meeting the
                   threshold or not. Must implement
                   org.apache.sqoop.validation.
ValidationThreshold Supported values: The value has to be a fully qualified class name.
Default value:     org.apache.sqoop.validation.AbsoluteValidationThreshold
```

Validation Failure Handler.

```
Property:          validation-failurehandler
Description:       Responsible for handling failures,
                   must implement
                   org.apache.sqoop.validation.
ValidationFailureHandler Supported values: The value has to be a fully qualified class name.
Default value:     org.apache.sqoop.validation.AbortOnFailureHandler
```

11.5。局限性

验证当前仅验证从单个表复制到HDFS的数据。以下是当前的限制：

- all-tables选项
- free-form 查询选项
- 数据导入到Hive，HBase或Accumulo中
- 使用–where参数导入表
- 增量导入

11.6。示例调用

命名表的基本进口 EMPLOYEES 的 corp 使用验证，验证行数数据库：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp  \     --table EMPLOYEES --validate
```

基本导出，用于填充 bar 启用了验证功能的表：

```
$ sqoop export --connect jdbc:mysql://db.example.com/foo --table bar  \     --export-dir /results/bar_data --validate
```

另一个覆盖验证参数的示例：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \     --validate --validator org.apache.sqoop.v
```

12.保存作业

可以通过多次发出同一命令来重复执行导入和导出。尤其是在使用增量导入功能时，这是预期的情况。

Sqoop允许您定义*保存的作业*，这使此过程更加容易。保存的作业记录以后执行Sqoop命令所需的配置信息。该 sqoop-job 工具的部分介绍了如何创建和使用已保存的作业。

默认情况下，职位描述保存到的专用存储库中 $HOME/.sqoop/ 。您可以将Sqoop配置为使用共享 *metastore*，这使保存的作业可供共享集群中的多个用户使用。 sqoop-metastore 工具上的部分介绍了如何启动Metastore 。

13 sqoop-job

13.1。目的

作业工具可让您创建并处理已保存的作业。保存的作业会记住用于指定作业的参数，因此可以通过按其句柄调用作业来重新执行它们。

如果将已保存的作业配置为执行增量导入，则有关已保存的作业中有关最近导入的行的状态将更新，以允许该作业仅连续地导入最新的行。

13.2。句法

```
$ sqoop job (generic-args) (job-args) [-[subtool-name] (subtool-args)]
$ sqoop-job (generic-args) (job-args) [-[subtool-name] (subtool-args)]
```

尽管Hadoop通用参数必须在任何作业参数之前，但作业参数可以相对于彼此以任何顺序输入。

表33.作业管理选项：

争论	描述
--create <job-id>	用指定的作业ID（名称）定义一个新的保存的作业。 -- 应该指定第二个Sqoop命令行，用a分隔；这定义了保存的作业。
--delete <job-id>	删除已保存的作业。
--exec <job-id>	给定一个用定义的作业 --create ，请运行保存的作业。
--show <job-id>	显示已保存作业的参数。
--list	列出所有已保存的作业

通过 --create 操作完成创建保存的作业。此操作需要 -- 后跟工具名称及其参数。该工具及其参数将构成已保存作业的基础。考虑：

```
$ sqoop job --create myjob-import --connect jdbc：mysql：//example.com/db \
    --table mytable
```

这将创建一个名为 myjob 稍后可以执行的作业。作业未运行。现在，该作业在已保存的作业列表中可用：

```
$ sqoop job --list
Available jobs:   myjob
```

我们可以通过以下 show 操作检查作业的配置：

```
$ sqoop job --show myjob
Job: myjob
Tool: import
Options:
----------------------------
direct.import = false
codegen.input.delimiters.record = 0
hdfs.append.dir = false
db.table = mytable
 ...
```

如果我们对此感到满意，则可以执行以下操作 exec ：

```
$ sqoop job --exec myjob
10/08/19 13:08:45 INFO tool.CodeGenTool: Beginning code generation ...
```

该 exec 操作允许您在后面提供参数，以覆盖已保存作业的参数 -- 。例如，如果将数据库更改为需要用户名，则可以使用以下命令指定用户名和密码：

```
$ sqoop job --exec myjob -- --username someuser -P
Enter password:
...
```

表34. Metastore连接选项：

争论	描述
--meta-connect <jdbc-uri>	指定用于连接到元存储的JDBC连接字符串

默认情况下，私有元存储库在中实例化 $HOME/.sqoop 。如果已使用该 sqoop-metastore 工具配置了托管元存储，则可以通过指定 --meta-connect 参数连接到它。这是一个JDBC连接字符串，就像用于连接到数据库以进行导入的字符串一样。

在中 conf/sqoop-site.xml ， 您可以 sqoop.metastore.client.autoconnect.url 使用此地址进行配置 ， 因此不必提供 --meta-connect 使用远程元存储库的权限。还可以修改此参数以将私有元存储移动到文件系统上的主目录以外的位置。

如果 sqoop.metastore.client.enable.autoconnect 使用该值进行配置 false ， 则必须显式提供 --meta-connect 。

表35.常用选项：

争论	描述
--help	打印使用说明
--verbose	在工作时打印更多信息

13.3。保存的作业和密码

Sqoop元存储库不是安全资源。多个用户可以访问其内容。因此，Sqoop不会将密码存储在metastore中。如果创建需要密码的作业，则每次执行该作业时都会提示您输入该密码。

您可以通过在配置中将设置 sqoop.metastore.client.record.password 为来启用元存储中的密码 true 。

请注意， 如果要通过Oozie执行保存的作业 sqoop.metastore.client.record.password ， true 则必须设置为，因为Sqoop在作为Oozie任务执行时无法提示用户输入密码。

13.4。保存作业和增量导入

通过将*检查列*中的值与最新导入的参考值进行比较来执行增量导入。例如，如果 --incremental append 指定了 参数以及 --check-column id 和 --last-value 100 ， id > 100 则将导入所有带有的行。如果从命令行运行增量导入，则应 --last-value 在后续增量导入中指定的值将被打印到屏幕上，以供您参考。如果从已保存的作业运行增量导入，则此值将保留在已保存的作业中。随后的运行 sqoop job --exec someIncrementalJob 将继续仅导入比以前导入的行更新的行。

14。 sqoop-metastore

14.1。目的

该 metastore 工具将Sqoop配置为托管共享元数据存储库。多个用户和/或远程用户可以定义和执行 sqoop job 此metastore中定义的保存的作业（使用创建的）。

必须将客户端配置为 sqoop-site.xml 在 --meta-connect 参数中或与参数一起连接到元存储。

14.2。句法

```
$ sqoop metastore (generic-args) (metastore-args)
$ sqoop-metastore (generic-args) (metastore-args)
```

尽管Hadoop通用参数必须在任何metastore参数之前，但是可以相对于彼此以任何顺序输入metastore参数。

表36. Metastore管理选项：

争论	描述
--shutdown	关闭同一台计算机上正在运行的Metastore实例。

运行 sqoop-metastore 将在当前计算机上启动一个共享的HSQLDB数据库实例。客户端可以连接到此metastore并创建可以在用户之间共享以执行的作业。

Metastore文件在磁盘上的位置由中的 sqoop.metastore.server.location 属性控制 conf/sqoop-site.xml 。这应该指向本地文件系统上的目录。

Metastore可通过TCP / IP使用。该端口由 sqoop.metastore.server.port 配置参数控制，默认为16000。

客户端应通过指定 sqoop.metastore.client.autoconnect.url 或 --meta-connect 使用值连接到元存储 jdbc:hsqldb:hsql://<server-name>:<port>/sqoop 。例如， jdbc:hsqldb:hsql://metaserver.example.com:16000/sqoop 。

该元存储库可以托管在Hadoop群集内的计算机上，也可以托管在网络上的其他位置。

15 sqoop-merge

15.1。目的

合并工具允许您合并两个数据集，其中一个数据集中的条目应覆盖较旧数据集的条目。例如，以上次修改模式运行的增量导入将在HDFS中生成多个数据集，其中每个数据集中会依次出现更新的数据。该 merge 工具会将两个数据集"拼合"为一个，并为每个主键获取最新的可用记录。

15.2。句法

```
$ sqoop merge (generic-args) (merge-args)
$ sqoop-merge (generic-args) (merge-args)
```

尽管Hadoop通用参数必须在任何合并参数之前，但作业参数可以相对于彼此以任何顺序输入。

表37.合并选项：

争论	描述
--class-name <class>	指定在合并作业期间要使用的特定于记录的类的名称。
--jar-file <file>	指定要从中加载记录类的jar的名称。
--merge-key <col>	指定用作合并键的列名。

争论	描述
--new-data <path>	指定较新数据集的路径。
--onto <path>	指定旧数据集的路径。
--target-dir <path>	指定合并作业输出的目标路径。

该 merge 工具运行MapReduce作业，该作业需要两个目录作为输入：较新的数据集和较旧的数据集。这些分别用 --new-data 和指定 --onto 。MapReduce作业的输出将放置在所指定的HDFS中的目录中 --target-dir 。

合并数据集时，假定每个记录中都有一个唯一的主键值。主键的列用指定 --merge-key 。同一数据集中的多个行不应具有相同的主键，否则可能会发生数据丢失。

要解析数据集并提取键列，必须使用先前导入的自动生成的类。你应该指定类名称和jar文件 --class-name 和 --jar-file 。如果不可用，则可以使用该 codegen 工具重新创建类。

合并工具通常是在最后修改日期模式（ sqoop import --incremental lastmodified ... ）增量导入后运行的。

假设执行了两个增量导入，其中一些较旧的数据位于名为HDFS的目录中， older 而较新的数据位于名为的HDFS目录中 newer ，则可以将它们合并，如下所示：

```
$ sqoop merge-新数据-更新到旧--target-dir已合并\
    --jar文件datatypes.jar-类名称Foo-合并键ID
```

这将运行MapReduce作业，其中 id 每行的列中的值用于连接行；在行 newer 数据集将优先使用到的行 older 数据集。

这可以与基于SequenceFile，Avro和基于文本的增量导入一起使用。较新的和较旧的数据集的文件类型必须相同。

16。 sqoop-codegen

16.1。目的

该 codegen 工具生成Java类，以封装和解释导入的记录。记录的Java定义是在导入过程中实例化的，但是也可以单独执行。例如，如果Java源丢失，则可以重新创建它。可以创建类的新版本，这些新版本在字段之间使用不同的分隔符，依此类推。

16.2。句法

```
$ sqoop codegen (generic-args) (codegen-args)
$ sqoop-codegen (generic-args) (codegen-args)
```

尽管Hadoop通用参数必须在任何codegen参数之前，但是可以相对于彼此以任何顺序输入codegen参数。

表38.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码

争论	描述
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

表39.代码生成参数：

争论	描述
--bindir <dir>	编译对象的输出目录
--class-name <name>	设置生成的类名称。这将覆盖 --package-name 。与结合使用时 --jar-file ，设置输入类别。
--jar-file <file>	禁用代码生成；使用指定的罐子
--outdir <dir>	生成代码的输出目录
--package-name <name>	将自动生成的类放在此包中
--map-column-java <m>	覆盖已配置列的从SQL类型到Java类型的默认映射。

表40.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符

争论	描述
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段：，行：\n 转义者：\ 可选地，封闭者：'
--optionally-enclosed-by <char>	设置一个字段包围字符

表41.输入解析参数：

争论	描述
--input-enclosed-by <char>	设置必填字段
--input-escaped-by <char>	设置输入转义字符
--input-fields-terminated-by <char>	设置输入字段分隔符
--input-lines-terminated-by <char>	设置输入的行尾字符
--input-optionally-enclosed-by <char>	设置一个字段包围字符

表42. Hive参数：

争论	描述
--hive-home <dir>	覆写 $HIVE_HOME
--hive-import	将表导入到Hive中（如果未设置，则使用Hive的默认定界符。）
--hive-overwrite	覆盖Hive表中的现有数据。
--create-hive-table	如果设置，则目标配置单元将使作业失败
	表存在。默认情况下，此属性为false。
--hive-table <table-name>	设置导入Hive时要使用的表名。

争论	描述
--hive-drop-import-delims	导入到Hive时，从字符串字段中 删除\ *n*，\ *r*和\ *01*。
--hive-delims-replacement	导入到Hive时，将字符串字段中的\ *n*，\ *r*和\ *01* 替换为用户定义的字符串。
--hive-partition-key	要分区的配置单元字段的名称被分片
--hive-partition-value <v>	用作此作业的分区键的字符串值导入到此作业中的蜂巢中。
--map-column-hive <map>	覆盖已配置列的从SQL类型到Hive类型的默认映射。如果在此参数中指定逗号，请使用URL编码的键和值，例如，使用DECIMAL（1%2C%201）而不是DECIMAL（1，1）。

如果将Hive自变量提供给代码生成工具，则Sqoop将生成一个包含HQL语句的文件，以创建表并加载数据。

16.3。示例调用

重新创建 employees 公司数据库表的记录解释代码：

```
$ sqoop codegen --connect jdbc:mysql://db.example.com/corp \
--table employees
```

17。 sqoop-create-hive-table

17.1。目的

该 create-hive-table 工具将基于先前导入到HDFS或计划导入的数据库表的表定义来填充Hive元存储。这样可以有效地执行“ --hive-import ”步骤， sqoop-import 而无需运行前面的导入。

如果数据已经加载到HDFS，则可以使用此工具完成将数据导入到Hive的管道。您也可以使用此工具创建Hive表。然后，可以在用户执行预处理步骤之后将数据导入并填充到目标中。

17.2。句法

```
$ sqoop create-hive-table（generic-args）（create-hive-table-args）
$ sqoop-create-hive-table（generic-args）（create-hive-table-args）
```

尽管Hadoop通用参数必须在所有create-hive-table参数之前，但可以相对于彼此以任何顺序输入create-hive-table参数。

表43.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息

争论	描述
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

表44. Hive参数：

争论	描述
--hive-home <dir>	覆写 $HIVE_HOME
--hive-overwrite	覆盖Hive表中的现有数据。
--create-hive-table	如果设置，则目标配置单元将使作业失败
	表存在。默认情况下，此属性为false。
--hive-table <table-name>	设置导入Hive时要使用的表名。
--table	从中读取定义的数据库表。

表45.输出行格式参数：

争论	描述
--enclosed-by <char>	设置必填字段的封闭字符
--escaped-by <char>	设置转义字符
--fields-terminated-by <char>	设置字段分隔符
--lines-terminated-by <char>	设置行尾字符
--mysql-delimiters	使用MySQL的默认定界符集：字段：，行：\n 转义者：\ 可选地，封闭者：'
--optionally-enclosed-by <char>	设置一个字段包围字符

请勿将封闭式或转义分隔符与用于导入到Hive的输出格式参数一起使用。Hive当前无法解析它们。

17.3。示例调用

在Hive中定义一个表，该表 emps 使用基于名为的数据库表的定义进行定义 employees：

```
$ sqoop create-hive-table --connect jdbc：mysql：//db.example.com/corp \
    --table员工--hive-table emps
```

18岁 sqoop-eval

18.1。目的

该 eval 工具允许用户针对数据库快速运行简单的SQL查询。结果将打印到控制台。这使用户可以预览其导入查询，以确保他们导入所需的数据。

警告 该 eval 工具仅用于评估目的。您可以使用它来从Sqoop内验证数据库连接或测试简单查询。不应在生产工作流中使用它。

18.2。句法

```
$ sqoop eval (generic-args) (eval-args)
$ sqoop-eval (generic-args) (eval-args)
```

尽管Hadoop通用参数必须在任何eval参数之前，但可以相对于彼此以任何顺序输入eval参数。

表46.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

表47. SQL评估参数：

争论	描述
-e,--query <statement>	*statement* 在SQL中 执行。

18.3。示例调用

从 employees 表中选择十个记录：

```
$ sqoop eval --connect jdbc:mysql://db.example.com/corp \
--query "SELECT * FROM employees LIMIT 10"
```

在 foo 表格中插入一行：

```
$ sqoop eval --connect jdbc:mysql://db.example.com/corp \
-e "INSERT INTO foo VALUES(42, 'bar')"
```

19 sqoop-list-databases

19.1。目的

列出服务器上存在的数据库架构。

19.2。句法

```
$ sqoop list-databases（generic-args） （list-databases-args）
$ sqoop-list-databases（generic-args） （list-databases-args）
```

尽管Hadoop通用参数必须在任何list-database参数之前，但是list-database参数可以相对于彼此以任何顺序输入。

表48.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

19.3。示例调用

列出MySQL服务器上可用的数据库模式：

```
$ sqoop list-databases --connect jdbc:mysql://database.example.com/ information_schema employees
```

笔记 这仅适用于HSQLDB，MySQL和Oracle。与Oracle一起使用时，连接到数据库的用户必须具有DBA特权。

20 sqoop-list-tables

20.1。目的

列出数据库中存在的表。

20.2。句法

```
$ sqoop list-tables (generic-args) (list-tables-args)
$ sqoop-list-tables (generic-args) (list-tables-args)
```

尽管Hadoop通用参数必须在任何list-tables参数之前，但list-tables参数可以相对于彼此以任何顺序输入。

表49.常用参数

争论	描述
--connect <jdbc-uri>	指定JDBC连接字符串
--connection-manager <class-name>	指定要使用的连接管理器类
--driver <class-name>	手动指定要使用的JDBC驱动程序类
--hadoop-mapred-home <dir>	覆写$ HADOOP_MAPRED_HOME
--help	打印使用说明
--password-file	设置包含验证密码的文件的路径
-P	从控制台读取密码
--password <password>	设置认证密码

争论	描述
--username <username>	设置身份验证用户名
--verbose	在工作时打印更多信息
--connection-param-file <filename>	提供连接参数的可选属性文件
--relaxed-isolation	将连接事务隔离设置为读取未提交的映射器。

20.3。示例调用

列出“ corp”数据库中可用的表：

```
$ sqoop list-tables --connect jdbc:mysql://database.example.com/corp employees payroll_checks job_descriptions office_s
```

在使用postgresql的情况下，带有公共参数的列表表命令仅获取“公共”模式。对于自定义模式，请使用–schema参数列出特定模式的表。

```
$ sqoop list-tables --connect jdbc:postgresql://localhost/corp  --username name -P -- --schema payrolldept employees ex
```

21 sqoop-help

21.1。目的

列出Sqoop中可用的工具并说明其用法。

21.2。句法

```
$ sqoop help [工具名称]
$ sqoop-help [工具名称]
```

如果未提供工具名称（例如，用户运行 sqoop help ），则会列出可用的工具。使用工具名称时，控制台上会显示该特定工具的使用说明。

21.3。示例调用

列出可用的工具：

```
$ sqoop帮助
用法：sqoop COMMAND [ARGS]

可用命令：
  codegen            生成代码以与数据库记录进行交互
  create-hive-table  将表定义导入Hive
  eval               评估SQL语句并显示结果
  export             将HDFS目录导出到数据库表

...

有关特定命令的信息，请参见“sqoop help COMMAND”。
```

显示该 import 工具的使用说明：

```
$ bin / sqoop帮助导入
用法：sqoop导入[GENERIC-ARGS] [TOOL-ARGS]

常用参数：
   --connect <jdbc-uri>  指定JDBC连接字符串
   --connection-manager <类名称>  指定要使用的连接管理器类
   --driver <class-name>  手动指定要使用的JDBC驱动程序类
   --hadoop-mapred-home <dir>  覆盖$ HADOOP_MAPRED_HOME
   --help  打印用法说明
   --password-file  设置包含认证密码的文件的路径
   -P  从控制台读取密码
   --password <密码>  设置身份验证密码
```

```
   --username <用户名>   设置身份验证用户名
   --verbose   在工作时打印更多信息
   --hadoop-home <dir>   已弃用。覆写$ HADOOP_HOME

导入控制参数：
   --as-avrodatafile   将数据导入到Avro数据文件
   --as-sequencefile   将数据导入到SequenceFiles
   --as-textfile   将数据导入为纯文本（默认）
   --as-parquetfile   将数据导入Parquet数据文件
...
```

22 sqoop-version

22.1 目的

显示Sqoop的版本信息。

22.2。句法

22.3。示例调用

显示版本：

```
$ sqoop version
Sqoop {revnumber}
git commit id 46b3e06b79a8411320d77c984c3030db47dd1c22 Compiled by aaron@jargon on Mon May 17 13:43:22 PDT 2010
```

23. Sqoop-HCatalog集成

23.1。HCatalog背景

HCatalog是Hadoop的表和存储管理服务，它使使用不同数据处理工具Pig，MapReduce和Hive的用户可以更轻松地在网格上读取和写入数据。HCatalog的表抽象为用户提供了Hadoop分布式文件系统（HDFS）中数据的关系视图，并确保用户不必担心数据的存储位置或格式：RCFile格式，文本文件或SequenceFiles。

HCatalog支持以已写入Hive SerDe（serializer-deserializer）的任何格式读取和写入文件。默认情况下，HCatalog支持RCFile，CSV，JSON和SequenceFile格式。要使用自定义格式，必须提供InputFormat和OutputFormat以及SerDe。

HCatalog具有抽象各种存储格式的功能，用于为Sqoop提供RCFile（和将来的文件类型）支持。

23.2。将HCatalog表暴露给Sqoop

HCatalog与Sqoop的集成是在支持Avro和Hive表的现有功能集上进行的。引入了七个新的命令行选项，并且为Hive定义的一些命令行选项已被重用。

23.2.1。新的命令行选项

`--hcatalog-database` 指定HCatalog表的数据库名称。如果未指定， `default` 则使用默认数据库名称。提供 `--hcatalog-database` 不带选项 `--hcatalog-table` 是错误的。这不是必需的选项。 `--hcatalog-table` 此选项的参数值为HCatalog表名。该 `--hcatalog-table` 选项的存在表示导入

或导出作业是使用HCatalog表完成的，并且是HCatalog作业的必需选项。 --hcatalog-home HCatalog安装的主目录。该目录应包含一个 lib 子目录和一个 share/hcatalog 带有必需HCatalog库的子目录。如果未指定， hcatalog.home 将检查系统属性，否则， HCAT_HOME 将检查系统环境变量。如果未设置这些设置，则将使用默认值，并且当前默认设置为 /usr/lib/hcatalog 。这不是必需的选项。 --create-hcatalog-table 此选项指定在导入数据时是否应自动创建HCatalog表。默认情况下，假定HCatalog表存在。表名将与转换为小写的数据库表名相同。在 Automatic Table Creation 下面进一步描述。 --drop-and-create-hcatalog-table 与相同 --create-hcatalog-table ，但 drop if exists 在创建表之前执行。 --hcatalog-storage-stanza 此选项指定要附加到表的存储节。在 Automatic Table Creation 下面进一步描述。 --hcatalog-partition-keys 和 --hcatalog-partition-values 这两个选项用于指定多个静态分区键/值对。在以前的版本中， --hive-partition-key 和 --hive-partition-value 选项用于指定静态分区键/值对，但是只能提供一个级别的静态分区键。选项 --hcatalog-partition-keys 和 --hcatalog-partition-values 允许将多个键和值提供为静态分区键。多个选项值之间用，（逗号）分隔。

例如，如果要从中导出/导入的表的配置单元分区键是用分区键名year，month和date定义的，并且特定的分区是year = 1999，month = 12，day = 31，则是所需的分区，则这两个选项的值如下：

- --hcatalog-partition-keys 年月日
- --hcatalog-partition-values 1999,12,31

为了提供向后兼容性，如果未提供 --hcatalog-partition-keys 或 --hcatalog-partition-values 选项，则将使用 --hive-partitition-key 和（ --hive-partition-value 如果提供）。

仅指定 --hcatalog-partition-keys 或 --hcatalog-partition-values 选项之一是错误的。要么提供两个选项，要么都不提供选项。

23.2.2。支持的Sqoop Hive选项

以下Sqoop选项也与该 --hcatalog-table 选项一起使用，可为HCatalog作业提供其他输入。某些现有的Hive导入作业选项可与HCatalog作业一起重用，而不是出于相同目的创建特定于HCatalog的选项。 --map-column-hive 此选项将数据库列映射到具有特定HCatalog类型的HCatalog。 --hive-home Hive的家庭位置。 --hive-partition-key 用于静态分区过滤器。分区键的类型应为STRING。只能有一个静态分区

键。请参阅有关 --hcatalog-partition-keys 和 --hcatalog-partition-values 选项的讨论。 --hive-partition-value 与分区关联的值。请参阅有关 --hcatalog-partition-keys 和 --hcatalog-partition-values 选项的讨论。

23.2.3。直接模式支持

Sqoop中的HCatalog集成已得到增强，以支持直接模式连接器（这是特定于数据库的高性能连接器）。Netezza直接模式连接器已得到增强，可以利用此功能。

重要 当前仅启用Netezza直接模式连接器才能与HCatalog一起使用。

23.2.4。不支持的Sqoop选项

23.2.4.1。不支持的Sqoop Hive导入选项23.2.4.2。不支持的Sqoop导出和导入选项

23.2.4.1。不支持的Sqoop Hive导入选项

HCatalog作业不支持以下Sqoop Hive导入选项。

- --hive-import
- --hive-overwrite

23.2.4.2。不支持的Sqoop导出和导入选项

HCatalog作业不支持以下Sqoop导出和导入选项。

- --export-dir
- --target-dir
- --warehouse-dir
- --append
- --as-sequencefile
- --as-avrodatafile
- --as-parquetfile

23.2.5。忽略的Sqoop选项

HCatalog作业将忽略以下选项。

- 所有输入定界符选项都将被忽略。
- 除非使用 --hive-drop-import-delims 或，否则通常会忽略输出定界符 --hive-delims-replacement 。当指定了 --hive-drop-import-delims or --hive-delims-replacement 选项时，所有 CHAR 类型数据库表列都将进行后处理，以分别删除或替换定界符。见 Delimited Text Formats and Field and Line Delimiter Characters 下文。仅当HCatalog表使用文本格式时才需要。

23.3。自动创建表格

Sqoop的主要功能之一是在导入Hadoop时管理和创建表元数据。HCatalog导入作业还提供带有此功能的选项 --create-hcatalog-table 。此外，HCatalog集成的重要好处之一是可以为Sqoop数据移动作业提供存储不可知性。为了提供该功能，HCatalog导入作业提供了一个选项，该选项使用户可以为创建的表指定存储格式。

该选项 --create-hcatalog-table 用作指示必须在HCatalog导入作业中创建表的指示符。如果 --create-hcatalog-table 指定了该选项 并且该表存在，则表创建将失败并且作业将中止。

该选项 --hcatalog-storage-stanza 可用于指定新创建的表的存储格式。此选项的默认值为 stored as rcfile 。假定为此选项指定的值是有效的Hive存储格式表达式。 create table 作为自动创建表的一部分，它将被附加到HCatalog导入作业所生成的命令中。存储节中的任何错误都将导致表创建失败，并且导入作业将被中止。

支持在选项中引用的存储格式所需的任何额外的资源 --hcatalog-storage-stanza 应通过将其放置在提供给工作要么 $HIVE_HOME/lib 或通过为他们提供 HADOOP_CLASSPATH 和 LIBJAR 文件。

如果 --hive-partition-key 指定了该选项，则此选项的值将用作新创建表的分区键。此选项只能指定一个分区键。

当映射到HCatalog表时，对象名称将映射为以下指定的小写字母。这包括表名（与转换为小写的外部存储表名相同）和字段名。

23.4。分隔的文本格式以及字段和行分隔符

HCatalog支持使用分隔的文本格式作为表存储格式之一。但是，当使用定界文本并且导入的数据具有包含那些定界符的字段时，Hive可能会将数据解析为不同数量的字段和记录，从而失去数据保真度。

对于这种情况，可以使用以下现有的Sqoop导入选项之一：

- --hive-delims-replacement
- --hive-drop-import-delims

如果提供了这些选项中的任何一个用于导入，则将使用Hive分隔符处理来格式化STRING类型的任何列，然后将其写入HCatalog表。

23.5。HCatalog表要求

如果默认表创建选项（带有可选的存储节）不足，则应在将HCatalog表用作Sqoop作业之前创建HCatalog表。HCatalog支持的所有存储格式均可用于HCatalog表的创建。这使得此功能可以很容易地采用Hive项目中引入的新存储格式，例如ORC文件。

23.6。支持分区

Sqoop HCatalog功能支持以下表类型：

- 未分区表
- 已指定静态分区键的分区表
- 具有数据库结果集中的动态分区键的分区表
- 结合了静态键和其他动态分区键的分区表

23.7。模式映射

Sqoop当前不支持列名映射。但是，允许用户覆盖类型映射。类型映射松散地遵循Sqoop中已经存在的Hive类型映射，除了SQL类型FLOAT和REAL被映射为HCatalog类型float。在Hive的Sqoop类型映射中，这两个映射为double。类型映射主要仅用于检查列定义的正确性，并且可以使用–map-column-hive选项覆盖。

除二进制以外的所有类型都可以分配给String类型。

在导出和导入期间，任何数字类型的字段（int，shortint，tinyint，bigint和bigdecimal，float和double）都可以分配给任何数字类型的另一个字段。根据目标分配类型的精度和规模，可能会发生截断。

此外，日期/时间/时间戳映射到日期/时间戳配置单元类型。（完整的日期/时间/时间戳表示）。日期/时间/时间标记列也可以映射为bigint Hive类型，在这种情况下，该值将是自历元以来的毫秒数。

仅导入支持BLOB和CLOB。导入时，BLOB / CLOB对象以特定于Sqoop的格式存储，并且需要这种格式的知识才能在Pig / Hive作业或另一个Map Reduce作业中处理这些对象。

当映射到HCatalog字段时，数据库列名称将映射为它们的小写等效项。当前，不支持区分大小写的数据库对象名称。

受表约束，允许将一组列从表投影到HCatalog表或加载到列投影。将数据导入HCatalog表时，动态分区列（如果有的话）必须是投影的一部分。

动态分区字段应映射到使用NOT NULL属性定义的数据库列（尽管在架构映射过程中不会强制执行此操作）。在动态分区列的导入过程中，如果为空值将中止Sqoop作业。

23.8。支持HCatalog数据类型

支持Hive 0.13版本中的所有原始Hive类型。当前不支持所有复杂的HCatalog类型。

仅导入支持BLOB / CLOB数据库类型。

23.9。为Sqoop工作提供Hive和HCatalog库

在Sqoop中添加了对HCatalog的支持后，任何HCatalog作业都依赖于Sqoop客户端主机以及Map / Reduce任务运行位置上可用的一组jar文件。要运行HCatalog作业， HADOOP_CLASSPATH 必须在启动Sqoop HCatalog作业之前按如下所示设置环境变量 。

```
HADOOP_CLASSPATH=$(hcat -classpath)  export HADOOP_CLASSPATH
```

必要的HCatalog依赖关系将由Sqoop作业自动复制到分布式缓存。

23.10。例子

创建一个HCatalog表，例如：

```
hcat -e "create table txn(txn_date string, cust_id string, amount float, store_id int) partitioned by (cust_id string) stored as rcfile;"
```

然后可以按以下方式调用“ txn” HCatalog表的Sqoop导入和导出：

23.11。进口

```
$SQOOP_HOME/bin/sqoop import --connect <jdbc-url> -table <table-name> --hcatalog-table txn <other sqoop options>
```

23.12。出口

```
$SQOOP_HOME/bin/sqoop export --connect <jdbc-url> -table <table-name> --hcatalog-table txn <other sqoop options>
```

24.兼容性说明

Sqoop使用JDBC连接到数据库，并尽可能遵守已发布的标准。对于不支持符合标准的SQL的数据库，Sqoop使用备用代码路径来提供功能。通常，Sqoop被认为可以与大量数据库兼容，但是只有少数几个经过了测试。

尽管如此，在实施Sqoop时仍做出了一些特定于数据库的决定，并且某些数据库提供了对标准的扩展的其他设置。

本节描述了使用Sqoop测试的数据库，相对于规范而言Sqoop处理每个数据库的任何异常以及Sqoop中可用的任何特定于数据库的设置。

24.1。支持的数据库

尽管JDBC是一个兼容层，它使程序可以通过通用API访问许多不同的数据库，但是每个数据库使用的SQL语言略有不同可能意味着Sqoop不能开箱即用地使用每个数据库，或者某些数据库可能以低效的方式使用。

当您向Sqoop提供连接字符串时，它将检查协议方案，以确定要使用的适当的特定于供应商的逻辑。如果Sqoop知道给定的数据库，它将自动运行。否则，您可能需要指定要通过加载的驱动程序类 `--driver` 。这将使用通用代码路径，该路径将使用标准SQL访问数据库。Sqoop为某些数据库提供了更快的，非基于JDBC的访问机制。可以通过指定 `--direct` 参数来启用它们。

Sqoop包括对以下数据库的特定于供应商的支持：

数据库	版本	--direct 支持?	连接字符串匹配
数据库	1.8.0+	不	jdbc:hsqldb:*//
MySQL	5.0+	是的	jdbc:mysql://
Oracle	10.2.0+	是的	jdbc:oracle:*//
PostgreSQL	8.3+	是（仅导入）	jdbc:postgresql://
Cubrid	9.2+	不	jdbc:cubrid:*

Sqoop可以使用列出的数据库的旧版本，但是我们仅使用上面指定的版本对其进行了测试。

即使Sqoop内部支持数据库，您仍可能需要 $SQOOP_HOME/lib 在客户端路径上安装数据库供应商的JDBC驱动程序。Sqoop可以从 $SQOOP_HOME/lib 客户端上的任何jar中加载类，并将其用作其运行的任何MapReduce作业的一部分；与旧版本不同，您不再需要在服务器上的Hadoop库路径中安装JDBC jar。

24.2。MySQL

JDBC驱动程序：MySQL Connector / J

MySQL v5.0及更高版本对Sqoop进行了非常全面的介绍。Sqoop已通过测试 mysql-connector-java-5.1.13-bin.jar 。

24.2.1。zeroDateTimeBehavior

zeroDateTimeBehavior，即MySQL允许 '0000-00-00\' for DATE 列的值，这是SQL的非标准扩展。通过JDBC进行通信时，这些值以三种不同的方式之一进行处理：

- 转换为 NULL 。
- 在客户端中引发异常。
- 四舍五入到最接近的法定日期（ '0001-01-01\' ）。

您可以通过使用 zeroDateTimeBehavior 连接字符串的属性来指定行为。如果 zeroDateTimeBehavior 未指定属性，则Sqoop将使用该 convertToNull 行为。

您可以覆盖此行为。例如：

```
$ sqoop import --table foo \
    --connect jdbc:mysql://db.example.com/someDb?zeroDateTimeBehavior=round
```

24.2.2。 UNSIGNED 列

UNSIGNED 在MySQL中类型为的列可以保存0到2 ^ 32（ 4294967295 ）之间的值，但是数据库会将数据类型报告给Sqoop as INTEGER ，该数据类型可以保存在 -2147483648 和 之间的值 \+2147483647 。Sqoop当前无法导入 UNSIGNED 高于的值 2147483647 。

24.2.3。 BLOB 和 CLOB 列

Sqoop的直接模式不支持进口 BLOB ， CLOB 或 LONGVARBINARY 列。对这些列使用基于JDBC的导入。不要将 --direct 参数提供给导入工具。

24.2.4。以直接模式导入视图

Sqoop当前不支持直接模式下从视图导入。如果您需要导入视图（只需省略 --direct 参数），请使用基于JDBC的（非直接）模式。

24.3。PostgreSQL的

Sqoop支持PostgreSQL的基于JDBC的连接器：http 😔/jdbc.postgresql.org/

该连接器已经在PostgreSQL服务器9.1上使用JDBC驱动程序版本“ 9.1-903 JDBC 4”进行了测试。

24.3.1。以直接模式导入视图

Sqoop当前不支持直接模式下从视图导入。如果您需要导入视图（只需省略 --direct 参数），请使用基于JDBC的（非直接）模式。

24.4。Oracle

JDBC驱动程序： Oracle JDBC Thin驱动程序-Sqoop与兼容 ojdbc6.jar 。

Sqoop已通过Oracle 10.2.0 Express Edition进行了测试。Oracle与ANSI标准及其非标准JDBC驱动程序的SQL处理方法不同。因此，几个功能的工作方式有所不同。

24.4.1。日期和时间

Oracle JDBC表示 DATE ， TIME SQL类型表示为 TIMESTAMP 值。 DATE Oracle数据库中的任何列都将作为 TIMESTAMP Sqoop中的导入，并且Sqoop生成的代码会将这些值存储在 java.sql.Timestamp 字段中。

将数据导出回数据库时，Sqoop会将文本字段解析为 TIMESTAMP 类型（具有形式 yyyy-mm-dd HH:MM:SS.ffffffff ），即使您希望这些字段使用JDBC日期转义格式格式化 yyyy-mm-dd 。导出到Oracle的日期应格式化为完整时间戳。

Oracle还包括其他日期/时间类型 TIMESTAMP WITH TIMEZONE 和 TIMESTAMP WITH LOCAL TIMEZONE 。要支持这些类型，必须指定用户的会话时区。默认情况下，Sqoop将为 "GMT" Oracle指定时区。您可以通过 oracle.sessionTimeZone 在运行Sqoop作业时在命令行上指定Hadoop属性来覆盖此设置。例如：

```
$ sqoop import -D oracle.sessionTimeZone =美国/洛杉矶
    --connect jdbc：oracle：thin：@ // db.example.com/foo --table栏
```

请注意，Hadoop参数（ -D ... ）是*通用参数*，并且必须出现在特定于工具的参数（ --connect ， --table 等）之前。

有关会话时区字符串的合法值，请参见 http://download-west.oracle.com/docs/cd/B19306_01/server.102/b14225/applocaledata.htm#i637736。

24.5。Hive中的架构定义

Hive用户将注意到，SQL类型和Hive类型之间没有一对一的映射。在没有直接映射一般情况下，SQL类型（例如 DATE ， TIME 和 TIMESTAMP ）将被强制为 STRING 在蜂巢。在 NUMERIC 与 DECIMAL SQL类型会被裹挟 DOUBLE 。在这些情况下，Sqoop将在其日志消息中发出警告，通知您精度降低。

24.6。Hive中的Parquet支持

为了从MapReduce作业联系Hive MetaStore，将获取并传递委托令牌。必须正确设置HIVE_CONF_DIR和HIVE_HOME，才能将Hive添加到运行时类路径。否则，以Parquet格式导入/导出到Hive可能不起作用。

24.7。CUBRID

Sqoop支持基于JDBC的Cubrid连接器：http
😔
/www.cubrid.org/?mid = downloads&item = jdbc_driver

已使用带有Cubrid 9.2的JDBC驱动程序版本“ JDBC-9.2.0.0155-cubrid.jar”对连接器进行了测试。

25.特定连接器的注意事项

25.1。MySQL JDBC连接器

本节包含特定于MySQL JDBC连接器的信息。

25.1.1。Upsert功能

MySQL JDBC连接器使用参数支持upsert功能 `--update-mode allowinsert` 。为了实现该功能，Sqoop使用了MySQL子句INSERT INTO...ON DUPLICATE KEY UPDATE。此子句不允许用户指定应使用哪些列来区分是更新现有行还是添加新行。相反，此子句依赖于表的唯一键（主键属于此集合）。MySQL将尝试插入新行，如果插入失败并出现重复的唯一键错误，它将改为更新相应的行。结果，Sqoop忽略了参数中指定的值 `--update-key` ，但是用户需要至少指定一个有效列才能打开更新模式本身。

25.2。MySQL直接连接器

MySQL Direct Connector允许使用 `mysqldump` 和 `mysqlimport` 工具功能（而不是SQL选择和插入）更快地向MySQL导入和导出。

要使用MySQL Direct连接器，请 `--direct` 为您的导入或导出作业指定参数。

例子：

```
$ sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES \
--direct
```

将附加参数传递给mysqldump：

```
$ sqoop import --connect jdbc:mysql://server.foo.com/db --table bar \
--direct -- --default-character-set=latin1
```

25.2.1。要求

实用程序 mysqldump 和 mysqlimport 应该存在于在所有节点上运行Sqoop命令的用户的外壳路径中。要以该用户身份验证所有节点的SSH并执行这些命令。如果您遇到错误，Sqoop也是如此。

25.2.2。局限性

- 当前，直接连接器不支持大对象列（BLOB和CLOB）的导入。
- 不支持导入到HBase和Accumulo
- 不支持导出数据时使用登台表
- 不支持导入视图

25.2.3。直接模式交易

为了提高性能，每个编写器将大约每32 MB导出的数据提交一次当前事务。您可以通过*在*任何特定*子*工具的参数*之前*指定以下参数来控制此操作： `-D sqoop.mysql.export.checkpoint.bytes=size` ，其中*size*是一个以字节为单位的值。设定*尺寸*为0以禁用中间检查点，但是导出的单个文件将继续彼此独立地提交。

有时，您需要使用Sqoop将大数据导出到实时MySQL群集中，该群集承受着高负载，可以为来自应用程序用户的随机查询提供服务。尽管可以使用登台表轻松解决导出过程中的数据一致性问题，但是仍然存在因繁重导出而导致的性能影响问题。

首先，专用于导入过程的MySQL资源会在主服务器和从服务器上影响实时产品的性能。其次，即使服务器可以在不影响性能的情况下处理导入（mysqlimport应该相对“便宜”），导入大表也可能导致集群中的严重复制滞后，从而导致数据不一致。

使用 -D sqoop.mysql.export.sleep.ms=time ，其中*时间*是一个以毫秒为单位的值，您可以在传输中指定的字节数后暂停导出过程，以使服务器在检查点之间放松，并使副本追上来 sqoop.mysql.export.checkpoint.bytes 。使用这两个参数的不同设置进行试验，以归档不会危及MySQL群集稳定性的导出速度。

> **重要** 请注意，这是在表格中Sqoop任何参数 -D parameter=value 是Hadoop的*通用参数*和任何工具，具体参数之前必须出现（例如 --connect ， --table 等）。不要忘记，只有 --direct 标志集支持这些参数。

25.3。Microsoft SQL连接器

25.3.1。额外的论点

Microsoft SQL连接器支持的所有其他参数的列表如下所示：

表50.支持的Microsoft SQL Connector附加参数：

争论	描述
+-身份插入	在导出插入之前，将IDENTITY_INSERT设置为ON。
--non-resilient	不要尝试恢复失败的导出操作。

争论	描述
--schema <name>	sqoop应该使用的方案名称。默认值为“ dbo”。
--table-hints <hints>	表暗示了Sqoop应该用于数据移动。

25.3.2。允许插入身份

您可以允许在具有标识的列上插入。例如：

```
$ sqoop export ... --export-dir custom_dir --table custom_table---identity-insert
```

25.3.3。非弹性操作

您可以覆盖默认值，并且在导出过程中不使用弹性操作。这样可以避免重试失败的操作。例如：

```
$ sqoop export ... --export-dir custom_dir --table custom_table--非弹性
```

25.3.4。模式支持

如果需要使用位于非默认模式中的表，则可以通过 --schema 参数指定模式名称。导入和导出作业均支持自定义架构。例如：

```
$ sqoop import ... --table custom_table---schema custom_schema
```

25.3.5。表提示

Sqoop在导入和导出作业中均支持表提示。表提示仅用于在Microsoft SQL Server之间移动数据的查询，但不能用于元数据查询。您可以在 --table-hints 参数中指定以逗号分隔的表提示列表 。例如：

```
$ sqoop import ... --table custom_table---table-提示NOLOCK
```

25.4。PostgreSQL连接器

25.4.1。额外的论点

PostgreSQL连接器支持的所有其他参数的列表如下所示：

表51.受支持的PostgreSQL额外参数：

争论	描述
--schema <name>	sqoop应该使用的方案名称。默认值为“ public”。

25.4.2。模式支持

如果您需要使用默认模式以外的其他模式中的表，则需要指定额外参数 --schema 。导入和导出作业均支持自定义架构（但是，可选的登台表必须与目标表位于相同的架构中）。调用示例：

```
$ sqoop import ... --table custom_table---schema custom_schema
```

25.5。PostgreSQL直接连接器

PostgreSQL Direct Connector允许更快地从PostgresSQL“ COPY”命令导入和导出。

要使用PostgreSQL Direct Connector，请 --direct 为您的导入或导出作业指定参数。

与直接模式一起从PostgreSQL导入时，可以在单个文件达到一定大小后将导入拆分为单独的文件。此大小限制由 --direct-split-size 参数控制。

直接连接器还提供了其他额外的参数：

表52.直接模式下受支持的其他PostgreSQL额外参数：

争论	描述
--boolean-true-string <str>	将用于编码列 true 值的字符串 boolean 。
	默认值为“ TRUE”。
--boolean-false-string <str>	将用于编码列 false 值的字符串 boolean 。
	默认值为“ FALSE”。

25.5.1。要求

psql 在所有节点上运行Sqoop命令的用户的Shell路径中应该存在实用程序。要以该用户身份验证所有节点的SSH并执行这些命令。如果您遇到错误，Sqoop也是如此。

25.5.2。局限性

- 当前，直接连接器不支持大对象列（BLOB和CLOB）的导入。
- 不支持导入到HBase和Accumulo
- 不支持导入视图

25.6。pg_bulkload连接器

25.6.1。目的

pg_bulkload连接器是用于将数据导出到PostgreSQL的直接连接器。该连接器使用 pg_bulkload。用户将从pg_bulkload的功能中受益，例如绕过共享缓冲区和WAL的快速导出，灵活的错误记录处理以及具有过滤器功能的ETL功能。

25.6.2。要求

pg_bulkload连接器需要以下条件才能执行导出作业：

- 该pg_bulkload 必须安装数据库服务器和所有从节点上。然后可在下载页面中找到适用于RedHat或CentOS的RPM 。
- 在PostgreSQL的JDBC 需要客户端节点上。
- 执行pg_bulkload需要PostgreSQL数据库的超级用户角色。

25.6.3。句法

使用 --connection-manager 选项指定连接管理器的类名。

```
$ sqoop export（generic-args）--connection-manager org.apache.sqoop.manager.PGBulkloadManager（export-args）
$ sqoop-export（generic-args）-连接管理器org.apache.sqoop.manager.PGBulkloadManager（export-args）
```

该连接器支持如下所示的导出参数。

表53.支持的导出控制参数：

争论	描述
--export-dir <dir>	HDFS导出的源路径
-m,--num-mappers <n>	使用*n*个地图任务并行导出
--table <table-name>	要填充的表
--input-null-string <null-string>	字符串列将被解释为null的字符串

通过Hadoop配置属性指定了用于pg_bulkload执行的其他配置，可以使用 -D <property=value> 选项指定。因为Hadoop配置属性是sqoop的通用参数，所以它必须位于任何导出控制参数之前。

表54.受支持的导出控制属性：

财产	描述
mapred.reduce.tasks	用于分阶段的化简任务的数量。defalt值为1。每个任务都在单个事务中暂存。
pgbulkload.bin	每个从属节点上安装的pg_bulkoad二进制文件的路径。
pgbulkload.check.constraints	指定是否在加载期间检查CHECK约束。默认值为“是”。
pgbulkload.parse.errors	在解析，编码，过滤，约束检查和数据类型转换期间导致错误的Ingored记录的最大数量。错误记录记录在PARSE BADFILE中。默认值为INFINITE。

财产	描述
pgbulkload.duplicate.errors	违反唯一约束的Ingored记录数。复制的记录记录在DB服务器上的DUPLICATE BADFILE中。默认值为INFINITE。
pgbulkload.filter	指定过滤器功能以转换输入文件中的每一行。请参阅pg_bulkload文档，以了解如何编写FILTER函数。
pgbulkload.clear.staging.table	指示可以删除登台表中存在的任何数据。

这是完整命令行的示例。

```
$ sqoop出口\
    -Dmapred.reduce.tasks = 2
    -Dpgbulkload.bin =“ / usr / local / bin / pg_bulkload” \
    -Dpgbulkload.input.field.delim = $'\ t'\
    -Dpgbulkload.check.constraints =“是” \
    -Dpgbulkload.parse.errors =“ INFINITE” \
    -Dpgbulkload.duplicate.errors =“ INFINITE” \
    --connect jdbc：postgresql：//pgsql.example.net：5432 / sqooptest \
    --connection-manager org.apache.sqoop.manager.PGBulkloadManager \
    --table test --username sqooptest --export-dir = / test -m 2
```

25.6.4。数据暂存

pg_bulkload连接器的导出作业的每个map任务都会动态创建自己的登台表。登台表的名称是根据目标表和任务尝试ID决定的。例如，“test”表的登台表的名称类似于 `test_attempt_1345021837431_0001_m_000000_0` 。

如果任务成功完成或映射任务失败，则暂存表将自动删除。当reduce任务失败时，将保留任务的登台表以进行手动重试，并且用户必须照顾好它。

25.7。Netezza连接器

25.7.1。额外的论点

Netezza Connector支持的所有其他参数的列表如下所示：

表55.受支持的Netezza额外参数：

争论	描述
--partitioned-access	每个映射器是作用于表的数据片的子集还是所有默认值（对于标准模式）为“ false”，对于直接模式为“ true”。
--max-errors	仅适用于直接模式导出。此选项指定在传输数据时每个映射器的错误阈值。如果遇到的错误数超过此阈值，则作业将失败。
	预设值为1。
--log-dir	仅适用于直接模式导出。指定Netezza外部表操作日志存储在hadoop文件系统上的目录。日志存储在此目录下，其中一个目录用于作业，子目录用于每个任务编号和尝试。默认值为用户主目录。nzlog和nzbad文件将位于
	（logdir）/ job-id / job-attempt-id。
--trunc-string	仅适用于直接模式导出。指定系统是否将字符串截断到声明的存储器并加载数据。默认情况下，字符串截断被报告为错误。
--ctrl-chars	仅适用于直接模式导出。指定是否允许控制字符（ASCII字符1-31）成为char / nchar / varchar / nvarchar列的一部分。默认为false。
--crin-string	仅适用于直接模式导出。指定是否允许将回车符（ASCII字符13）作为char / nchar / varchar / nvarchar列的一部分。请注意，使用此选项，CR不再可以成为记录定界符。默认为false。
--ignore-zero	仅适用于直接模式导出。指定是否应作为加载到char / nchar / varchar / nvarchar列中的数据的一部分来扫描和忽略NUL字符（ASCII char 0）。默认为false。

25.7.2。直接模式

Netezza connector supports an optimized data transfer facility using the Netezza external tables feature. Each map tasks of Netezza connector's import job will work on a subset of the Netezza partitions and transparently create and use an external table to transport data. Similarly, export jobs will use the external table to push data fast onto the NZ system. Direct mode does not support staging tables, upsert options etc.

Here is an example of complete command line for import using the Netezza external table feature.

```
$ sqoop import \
    --direct \
    --connect jdbc:netezza://nzhost:5480/sqoop \
    --table nztable \
    --username nzuser \
    --password nzpass \
    --target-dir hdfsdir
```

Here is an example of complete command line for export with tab as the field terminator character.

```
$ sqoop export \
    --direct \
    --connect jdbc:netezza://nzhost:5480/sqoop \
    --table nztable \
    --username nzuser \
    --password nzpass \
    --export-dir hdfsdir \
    --input-fields-terminated-by "\t"
```

25.7.3. Null string handling

Netezza direct connector supports the null-string features of Sqoop. The null string values are converted to appropriate external table options during export and import operations.

Table 56. Supported export control arguments:

Argument	Description
--input-null-string <null-string>	The string to be interpreted as null for string columns.
--input-null-non-string <null-string>	The string to be interpreted as null for non string columns.

In the case of Netezza direct mode connector, both the arguments must be left to the default values or explicitly set to the same value. Furthermore the null string value is restricted to 0-4 utf8 characters.

On export, for non-string columns, if the chosen null value is a valid representation in the column domain, then the column might not be loaded as null. For example, if the null string value is specified as “1”, then on export, any occurrence of “1” in the input file will be loaded as value 1 instead of NULL for int columns.

It is suggested that the null value be specified as empty string for performance and consistency.

Table 57. Supported import control arguments:

Argument	Description
--null-string <null-string>	The string to be interpreted as null for string columns.
--null-non-string <null-string>	The string to be interpreted as null for non string columns.

对于Netezza直接模式连接器，必须将两个参数都保留为默认值或显式设置为相同的值。此外，空字符串值限制为0-4个utf8字符。

导入时，对于非字符串列，在当前实现中选择的空值对于非字符列将忽略空值表示。例如，如果将空字符串值指定为“ \ N”，则在导入时，表中非字符列的任何出现的NULL都将作为空字符串而不是所选的空字符串表示形式\ N导入。

建议将空值指定为空字符串以提高性能和一致性。

25.8。适用于Oracle和Hadoop的数据连接器

25.8.1。关于

25.8.1.1。职位25.8.1.2。标准Oracle Manager如何用于导入25.8.1.3。Oracle和Hadoop的数据连接器如何用于导入25.8.1.4。适用于Oracle和Hadoop导出的数据连接器

Sqoop现在包含了适用于Oracle和Hadoop的数据连接器。

可以通过 --direct 为导入或导出作业指定参数来启用它。

25.8.1.1。职位

适用于Oracle和Hadoop的数据连接器将检查每个Sqoop作业，并对与Sqoop内置的Oracle管理器相比性能更好的作业承担责任。

适用于Oracle和Hadoop的数据连接器对以下Sqoop作业类型承担责任：

- **导入非增量**作业。
- **出口**工作

- 适用于Oracle和Hadoop的数据连接器不承担其他Sqoop作业类型的责任。例如，适用于Oracle和Hadoop的数据连接器不接受**评估**作业等。

适用于Oracle和Hadoop的数据连接器接受具有以下属性的那些Sqoop作业的责任：

- 与Oracle相关
- 基于表的作业-使用表参数且指定对象为表的作业。笔记适用于Oracle和Hadoop的数据连接器不会处理索引组织的表，除非对该表进行了分区并将 oraoop.chunk.method 其设置为 PARTITION
- 至少有2个映射器— Sqoop命令行不包括的作业： --num-mappers 1

25.8.1.2。标准Oracle Manager如何用于导入

Sqoop中内置的Oracle管理器为每个映射器使用基于范围的查询。每个映射器都会执行以下形式的查询：

```
选择*从某表中，id> = lo和id <hi
```

的LO和**喜**值是基于在表中被分割列映射器中的数据的最大值的数目以及最小和。

如果表上不存在合适的索引，则这些查询将导致Oracle内进行完整的表扫描。即使具有合适的索引，多个映射器也可能会获取存储在相同Oracle块中的数据，从而导致冗余的IO调用。

25.8.1.3。Oracle和Hadoop的数据连接器如何用于导入

用于Oracle和Hadoop的数据连接器为以下形式的映射器生成查询：

```
选择 *
  从某桌
 在rowid> = dbms_rowid.rowid_create（1，893，1，279，0）并且
```

```
        rowid <= dbms_rowid.rowid_create（1，893，1，286，32767）
```

用于Oracle和Hadoop的数据连接器查询可确保：

- 没有两个映射器从同一个Oracle块读取数据。这样可以最大程度地减少冗余IO。
- 该表不需要索引。
- Sqoop命令行不需要指定 --split-by 列。

25.8.1.4。适用于Oracle和Hadoop导出的数据连接器

适用于Oracle和Hadoop的数据连接器的优势：

- **合并导出工具**-通过修改更改的行并从HDFS文件中插入Oracle表中以前不存在的行来更新Oracle表。Oracle和Hadoop的Merge-Export连接器是唯一的-没有Sqoop等效项。
- **对Oracle数据库的影响较小**-更新Oracle表中已更改的行，而不是Oracle表中的所有行。这具有性能优势，并减少了查询对Oracle的影响（例如，Oracle重做日志）。
- **改进的性能**-使用分区表，映射器利用临时的Oracle表，该表允许并行插入和直接路径写入。

25.8.2。要求

25.8.2.1。确保正确设置了Oracle数据库JDBC驱动程序25.8.2.2。Oracle角色和特权25.8.2.3。导出所需的其他Oracle角色和特权25.8.2.4。支持的数据类型

25.8.2.1。确保正确设置了Oracle数据库JDBC驱动程序

您可能要确保在系统上正确设置了Oracle Database 11g第2版JDBC驱动程序。Sqoop需要此驱动程序才能与Oracle一起使用。

Oracle Database 11g第2版JDBC驱动程序文件为 ojdbc6.jar （3.2Mb）。

如果该文件不在您的系统上，则从以下位置下载它：http : //www.oracle.com/technetwork/database/features/jdbc/index-091264.html

该文件应放在 $SQOOP_HOME/lib 目录中。

25.8.2.2。Oracle角色和特权

用于Oracle和Hadoop的数据连接器的Oracle用户需要以下角色和特权：

- create session

此外，用户必须具有“选择任何词典”特权或“ select_catalog_role”角色或以下所有对象特权：

- select on v_$instance
- select on dba_tables
- select on dba_tab_columns
- select on dba_objects
- select on dba_extents
- select on dba_segments —仅对于Sqoop导入是必需的
- select on dba_constraints —仅对于Sqoop导入是必需的
- select on v_$database —仅对于Sqoop导入是必需的
- select on v_$parameter —仅对于Sqoop导入是必需的

	笔记

用户还需要alter session特权才能使用会话跟踪功能。有关更多信息，请参见“oraoop.oracle.session.initialization.statements”。	

25.8.2.3。导出所需的其他Oracle角色和特权

用于Oracle和Hadoop的数据连接器的Oracle用户要求：

- Oracle导出表所在的表空间上的配额。一个用于实现此目的的Oracle命令示例是更改表空间上的用户名限制
- 以下特权：出口种类所需特权全部出口 create table select on dba_tab_partitions select on dba_tab_subpartitions select on dba_indexes select on dba_ind_columns 将模板表插入-导出到另一个模式 select any table create any table insert any table alter any table （分区）不带模板表的插入-导出到另一个模式 select,insert on table （无分区） select,alter on table （分区）更新-导出到另一个模式 select,update on table （无分区） select,delete,alter,insert on table （分区）合并导出到另一个模式 select,insert,update on table （无分区） select,insert,delete,alter on table （分区）

25.8.2.4. Supported Data Types

The following Oracle data types are supported by the Data Connector for Oracle and Hadoop:

BINARY_DOUBLE	NCLOB
BINARY_FLOAT	NUMBER
BLOB	NVARCHAR2
CHAR	RAW
CLOB	ROWID
DATE	TIMESTAMP

FLOAT	TIMESTAMP WITH TIME ZONE
INTERVAL DAY TO SECOND	TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH	URITYPE
LONG	VARCHAR2
NCHAR	

All other Oracle column types are NOT supported. Example Oracle column types NOT supported by Data Connector for Oracle and Hadoop include:

All of the ANY types	BFILE
All of the MEDIA types	LONG RAW
All of the SPATIAL types	MLSLABEL
任何被称为UNDEFINED的类型	UROWID
所有自定义（用户定义）URI类型	XML类型

	笔记
适用于Oracle和Hadoop导入的数据连接器支持RAW，LONG和LOB（BLOB，CLOB和NCLOB）数据类型。Data Connector for Oracle和Hadoop导出不支持它们。	

25.8.3。使用适用于Oracle和Hadoop的数据连接器执行Sqoop

25.8.3.1。连接到Oracle / Oracle RAC

Sqoop --connect 参数定义要连接的Oracle实例或Oracle RAC。所有Sqoop导入和导出命令都需要它。

适用于Oracle和Hadoop的数据连接器期望关联的连接字符串具有特定的格式，具体取决于是否定义了Oracle SID，服务或TNS名称。基于TNS名称的URL方案可用于使用Oracle钱包启用身份验证。

```
--connect jdbc:oracle:thin:@OracleServer:OraclePort:OracleSID
```

```
--connect jdbc:oracle:thin:@//OracleServer:OraclePort/OracleService
```

```
--connect jdbc:oracle:thin:@TNSName
```

25.8.3.2。连接到Oracle数据库实例

参数/组件	描述
jdbc:oracle:thin	适用于Oracle和Hadoop的数据连接器要求连接字符串以jdbc：oracle开头。
	适用于Oracle和Hadoop的数据连接器已经通过瘦驱动程序进行了测试，但是它应该可以与其他驱动程序（例如OCI）同样良好地工作。
OracleServer	Oracle服务器的主机名。
OraclePort	连接到Oracle服务器的端口。
OracleSID	Oracle实例。

参数/组件	描述
OracleService	Oracle服务。
TNSName	条目的TNS名称，用于描述与Oracle服务器的连接。

	笔记
Hadoop映射器使用动态生成的JDBC URL连接到Oracle数据库。这是为了提高性能而设计的，但是可以通过指定以下内容来禁用它： -D oraoop.jdbc.url.verbatim=true	

25.8.3.3。连接到Oracle RAC

使用 --connect 上面的参数。连接字符串应指向Oracle RAC的一个实例。此Oracle实例的主机的侦听器将找到Oracle RAC的其他实例。

	笔记
为了提高性能，适用于Oracle和Hadoop的数据连接器标识了Oracle RAC的活动实例，并以循环方式将每个Hadoop映射器连接到它们。	

如果为此Oracle RAC定义了服务，则使用以下参数指定服务名称：

```
-D oraoop.oracle.rac.service.name=ServiceName
```

参数/组件	描述

参数/组件	描述
OracleServer:OraclePort:OracleInstance	命名一个Oracle RAC实例。Oracle和Hadoop的数据连接器为Oracle RAC的所有实例假定相同的端口号。
	此Oracle实例的主机的侦听器用于查找Oracle RAC的其他实例。有关更多信息，请在主机命令行上输入以下命令：
	lsnrctl status
-D oraoop.oracle.rac.service.name=ServiceName	在Oracle RAC中要连接的服务。
	将建立与与所提供的服务相关联的Oracle RAC的所有实例的连接 ServiceName 。
	如果省略，将建立到Oracle RAC的所有实例的连接。
	此Oracle实例的主机的侦听器需要知道 ServiceName Oracle RAC以及所有实例。有关更多信息，请在主机命令行上输入以下命令：
	lsnrctl status

25.8.3.4。登录到Oracle实例

在Sqoop命令行上登录到Oracle实例：

```
--connect jdbc:oracle:thin:@OracleServer:OraclePort:OracleInstance --username UserName -P
```

参数/组件	描述
--username UserName	The username to login to the Oracle instance (SID).
-P	You will be prompted for the password to login to the Oracle instance.

25.8.3.5. Kill Data Connector for Oracle and Hadoop Jobs

Use the Hadoop Job Tracker to kill the Sqoop job, just as you would kill any other Map-Reduce job.

```
$ hadoop job -kill jobid
```

To allow an Oracle DBA to kill a Data Connector for Oracle and Hadoop job (via killing the sessions in Oracle) you need to prevent Map-Reduce from re-attempting failed jobs. This is done via the following Sqoop command-line switch:

```
-D mapred.map.max.attempts=1
```

This sends instructions similar to the following to the console:

```
14/07/07 15:24:51 INFO oracle.OraOopManagerFactory:
Note: This Data Connector for Oracle and Hadoop job can be killed via Oracle
by executing the following statement:
  begin
    对于进入的行（从v $ session中选择sid，serial#，其中
                 module =“用于Oracle和Hadoop的数据连接器”和
                 action ='import 20140707152451EST'）循环
      立即执行'alter system kill session'''|| row.sid ||
                          ', '|| row.serial#|| '';
    结束循环；
  结尾；
```

25.8.4。从Oracle导入数据

25.8.4.1。将Hadoop文件与Oracle表分区匹配25.8.4.2。指定要导入的分区25.8.4.3。一致读取：所有映射器从同一时间点读取

执行Sqoop。以下是示例命令：

```
$ sqoop import --direct --connect ... --table OracleTableName
```

如果适用于Oracle和Hadoop的数据连接器接受了作业，则输出以下文本：

```
**************************************************
***使用适用于Oracle和Hadoop的数据连接器***
**************************************************
```

	笔记
有关该 --connect 参数的更多信息。有关更多信息，请参见“连接到Oracle / Oracle RAC”。如果Java内存不足，解决方法是指定每个映射器的JVM内存分配。添加以下参数，例如分配4GB： -Dmapred.child.java.opts=-Xmx4000M 默认情况下，SELECT语句中包含Oracle优化程序提示。有关更多信息，请参见“oraoop.import.hint”。您可以按以下方式更改命令行上的提示： -Doraoop.import.hint="NO_INDEX(t)" 您可以按如下所示关闭命令行上的提示（注意双引号之间的空格）： -Doraoop.import.hint=" "	

25.8.4.1。将Hadoop文件与Oracle表分区匹配

-Doraoop.chunk.method={ROWID|PARTITION}

要以某种方式从分区表中导入数据，以使Hadoop中生成的HDFS文件夹结构与表的分区相匹配，请将块方法设置为PARTITION。另一种（默认）块方法是ROWID。

	笔记
为了使Hadoop文件的数量与Oracle分区的数量相匹配，请将映射器的数量设置为	

大于或等于分区的数量。如果表未分区，则值PARTITION将导致错误。

25.8.4.2。指定要导入的分区

-Doraoop.import.partitions=PartitionA,PartitionB --table OracleTableName

进口 PartitionA 及 PartitionB 的 OracleTableName 。

	笔记
您可以将单个分区名称括在双引号中，以保留字母大小写或名称具有特殊字符的情况。 -Doraoop.import.partitions='"PartitionA",PartitionB' --table OracleTableName 如果分区名称不是双引号，则其名称将自动转换为大写字母，即上面的PARTITIONB。When using double quotes the entire list of partition names must be enclosed in single quotes.If the last partition name in the list is double quoted then there must be a comma at the end of the list. -Doraoop.import.partitions='"PartitionA","PartitionB",' --table OracleTableName Name each partition to be included. There is no facility to provide a range of partition names.There is no facility to define sub partitions. The entire partition is included/excluded as per the filter.	

25.8.4.3. Consistent Read: All Mappers Read From The Same Point In Time

-Doraoop.import.consistent.read={true|false}

When set to `false` (by default) each mapper runs a select query. This will return potentially inconsistent data if there are a lot of DML operations on the table at the time of import.

Set to `true` to ensure all mappers read from the same point in time. The System Change Number (SCN) is passed down to all mappers, which use the Oracle Flashback Query to query the table as at that SCN.

	Note
Values `true` | `false` are case sensitive.By default the SCN is taken from V$database. You can specify the SCN in the following command - `Doraoop.import.consistent.read.scn=12345`	

25.8.5. Export Data into Oracle

25.8.5.1. Insert-Export25.8.5.2. Update-Export25.8.5.3. Merge-Export25.8.5.4. Create Oracle Tables25.8.5.5. NOLOGGING25.8.5.6. Partitioning25.8.5.7. Match Rows Via Multiple Columns25.8.5.8. Storage Clauses

Execute Sqoop. Following is an example command:

```
$ sqoop export --direct --connect ... --table OracleTableName --export-dir /user/username/tablename
```

The Data Connector for Oracle and Hadoop accepts all jobs that export data to Oracle. You can verify The Data Connector for Oracle and Hadoop is in use by checking the following text is output:

```
**************************************************
*** Using Data Connector for Oracle and Hadoop ***
**************************************************
```

	Note

OracleTableName is the Oracle table the data will export into. OracleTableName can be in a schema other than that for the connecting user. Prefix the table name with the schema, for example SchemaName.OracleTableName .Hadoop tables are picked up from the /user/username/tablename directory.The export will fail if the Hadoop file contains any fields of a data type not supported by The Data Connector for Oracle and Hadoop. See “Supported Data Types” for more information.The export will fail if the column definitions in the Hadoop table do not exactly match the column definitions in the Oracle table.The Data Connector for Oracle and Hadoop indicates if it finds temporary tables that it created more than a day ago that still exist. Usually these tables can be dropped. The only circumstance when these tables should not be dropped is when an The Data Connector for Oracle and Hadoop job has been running for more than 24 hours and is still running.More information is available on the --connect parameter. See “Connect to Oracle / Oracle RAC” for more information.	

25.8.5.1. Insert-Export

Appends data to OracleTableName . It does not modify existing data in OracleTableName .

Insert-Export is the default method, executed in the absence of the --update-key parameter . All rows in the HDFS file in /user/UserName/TableName are inserted into OracleTableName . No change is made to pre-existing data in OracleTableName .

```
$ sqoop export --direct --connect ... --table OracleTableName --export-dir /user/username/tablename
```

	Note
If OracleTableName was previously created by The Data Connector for Oracle and Hadoop with partitions then this export will create a new partition for the	

data being inserted.When creating OracleTableName specify a template. See “Create Oracle Tables” for more information.	

25.8.5.2. Update-Export

--update-key OBJECT

Updates existing rows in OracleTableName .

Rows in the HDFS file in /user/UserName/TableName are matched to rows in OracleTableName by the OBJECT column. Rows that match are copied from the HDFS file to the Oracle table. No action is taken on rows that do not match.

$ sqoop export --direct --connect … --update-key OBJECT --table OracleTableName --export-dir /user/username/tablename

	Note
If OracleTableName was previously created by The Data Connector for Oracle and Hadoop with partitions then this export will create a new partition for the data being inserted. Updated rows will be moved to the new partition that was created for the export.For performance reasons it is strongly recommended that where more than a few rows are involved column OBJECT be an index column of OracleTableName .Ensure the column name defined with --update-key OBJECT is specified in the correct letter case. Sqoop will show an error if the letter case is incorrect.It is possible to match rows via multiple columns. See “Match Rows Via Multiple Columns” for more information.	

25.8.5.3. Merge-Export

`--update-key OBJECT -Doraoop.export.merge=true`

Updates existing rows in `OracleTableName`. Copies across rows from the HDFS file that do not exist within the Oracle table.

Rows in the HDFS file in `/user/UserName/TableName` are matched to rows in `OracleTableName` by the `OBJECT` column. Rows that match are copied from the HDFS file to the Oracle table. Rows in the HDFS file that do not exist in `OracleTableName` are added to `OracleTableName`.

`$ sqoop export --direct --connect ... --update-key OBJECT -Doraoop.export.merge=true --table OracleTableName --export-dir /user/username/tablename`

	Note
Merge-Export is unique to The Data Connector for Oracle and Hadoop. It is not a standard Sqoop feature.If `OracleTableName` was previously created by The Data Connector for Oracle and Hadoop with partitions, then this export will create a new partition for the data being inserted. Updated rows will be moved to the new partition that was created for the export.For performance reasons it is strongly recommended that where more than a few rows are involved column `OBJECT` be an index column of `OracleTableName`.Ensure the column name defined with `--update-key OBJECT` is specified in the correct letter case. Sqoop will show an error if the letter case is incorrect.It is possible to match rows via multiple columns. See “Match Rows Via Multiple Columns” for more information.	

25.8.5.4. Create Oracle Tables

-Doraoop.template.table=TemplateTableName

Creates OracleTableName by replicating the structure and data types of TemplateTableName . TemplateTableName is a table that exists in Oracle prior to executing the Sqoop command.

	Note
The export will fail if the Hadoop file contains any fields of a data type not supported by The Data Connector for Oracle and Hadoop. See “Supported Data Types” for more information.The export will fail if the column definitions in the Hadoop table do not exactly match the column definitions in the Oracle table.此参数特定于创建Oracle表。如果 OracleTableName Oracle中已经存在，则导出将失败。	

示例命令：

$ sqoop export --direct --connect.. --table OracleTableName --export-dir /user/username/tablename -Doraoop.template.table=TemplateTableName

25.8.5.5。无记录

-Doraoop.nologging=true

将NOLOGGING选项分配给 OracleTableName 。

NOLOGGING可以提高性能，但是您将无法备份该表。

25.8.5.6。分区

-Doraoop.partitioned=true

对表进行分区具有以下好处：

- 通过允许每个映射器使用直接路径写入将数据插入到单独的Oracle表中，可以提高导出速度。（随后执行alter table交换子分区SQL语句以将数据交换到导出表中。）
- 您可以有选择地查询或删除每个Sqoop导出作业插入的数据。例如，您可以通过从表中删除旧分区来删除旧数据。

分区值是执行Sqoop导出作业时的SYSDATE。

由用于Oracle和Hadoop的数据连接器创建的分区表包括模板表中不存在的以下列：

- oraoop_export_sysdate -这是执行Sqoop导出作业时的Oracle SYSDATE。创建的表将通过此列进行分区。
- oraoop_mapper_id -这是用于处理HDFS文件中的行的Hadoop映射器的ID。每个分区都由该列细分。存在此列仅是为了便于在导出过程中由每个映射器执行的交换子分区机制。
- oraoop_mapper_row -映射器/分区中的唯一行ID。

	笔记
如果表需要唯一的行ID，则可以由oraoop_export_sysdate，oraoop_mapper_id和oraoop_mapper_row组合而成。	

25.8.5.7。通过多列匹配行

-Doraoop.update.key.extra.columns="ColumnA,ColumnB"

与Update-Export和Merge-Export配合使用以匹配多个列。要匹配的第一列是 --update-key OBJECT 。要匹配其他列，请在此参数上指定这些列。

	笔记
列名的字母大小写在此参数上并不重要。用于匹配的所有列都应建立索引。索引的前三项应为 ColumnA， ColumnB 并且列在 --update-key -上指定，但是指定列 的顺序并不重要。	

25.8.5.8。储存条款

-Doraoop.temporary.table.storage.clause="StorageClause"

-Doraoop.table.storage.clause="StorageClause"

用于通过TABLESPACE或COMPRESS中的Oracle子句来自定义存储

-Doraoop.table.storage.clause applies to the export table that is created from -Doraoop.template.table . See “Create Oracle Tables” for more information. -Doraoop.temporary.table.storage.clause applies to all other working tables that are created during the export process and then dropped at the end of the export job.

25.8.6. Manage Date And Timestamp Data Types

25.8.6.1. Import Date And Timestamp Data Types from Oracle

This section lists known differences in the data obtained by performing an Data Connector for Oracle and Hadoop import of an Oracle table versus a native Sqoop import of the same table.

25.8.6.2. The Data Connector for Oracle and Hadoop Does Not Apply A Time Zone to DATE / TIMESTAMP Data Types

Data stored in a DATE or TIMESTAMP column of an Oracle table is not associated with a time zone. Sqoop without the Data Connector for Oracle and Hadoop inappropriately applies time zone information to this data.

Take for example the following timestamp in an Oracle DATE or TIMESTAMP column: 2am on 3rd October, 2010 .

Request Sqoop without the Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. The data is adjusted to Melbourne Daylight Saving Time. The data is imported into Hadoop as: 3am on 3rd October, 2010.

The Data Connector for Oracle and Hadoop does not apply time zone information to these Oracle data-types. Even from a system located in Melbourne Australia, The Data Connector for Oracle and Hadoop ensures the Oracle and Hadoop timestamps match. The Data Connector for Oracle and Hadoop correctly imports this timestamp as: 2am on 3rd October, 2010 .

	Note
In order for The Data Connector for Oracle and Hadoop to ensure data accuracy,	

Oracle DATE and TIMESTAMP values must be represented by a String, even when `--as-sequencefile` is used on the Sqoop command-line to produce a binary file in Hadoop.

25.8.6.3. The Data Connector for Oracle and Hadoop Retains Time Zone Information in TIMEZONE Data Types

Data stored in a TIMESTAMP WITH TIME ZONE column of an Oracle table is associated with a time zone. This data consists of two distinct parts: when the event occurred and where the event occurred.

When Sqoop without The Data Connector for Oracle and Hadoop is used to import data it converts the timestamp to the time zone of the system running Sqoop and omits the component of the data that specifies where the event occurred.

Take for example the following timestamps (with time zone) in an Oracle TIMESTAMP WITH TIME ZONE column:

```
2:59:00 am on 4th April, 2010. Australia/Melbourne
2:59:00 am on 4th April, 2010. America/New York
```

Request Sqoop without The Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. From the data imported into Hadoop we know when the events occurred, assuming we know the Sqoop command was run from a system located in the Australia/Melbourne time zone, but we have lost the information regarding where the event occurred.

```
2010-04-04 02:59:00.0
2010-04-04 16:59:00.0
```

Sqoop with The Data Connector for Oracle and Hadoop imports the example timestamps as follows. The Data Connector for Oracle and Hadoop retains the time zone portion of the data.

```
2010-04-04 02:59:00.0 Australia/Melbourne
2010-04-04 02:59:00.0 America/New_York
```

25.8.6.4. Data Connector for Oracle and Hadoop Explicitly States Time Zone for LOCAL TIMEZONE Data Types

Data stored in a TIMESTAMP WITH LOCAL TIME ZONE column of an Oracle table is associated with a time zone. Multiple end-users in differing time zones (locales) will each have that data expressed as a timestamp within their respective locale.

When Sqoop without the Data Connector for Oracle and Hadoop is used to import data it converts the timestamp to the time zone of the system running Sqoop and omits the component of the data that specifies location.

Take for example the following two timestamps (with time zone) in an Oracle TIMESTAMP WITH LOCAL TIME ZONE column:

```
2:59:00 am on 4th April, 2010. Australia/Melbourne
2:59:00 am on 4th April, 2010. America/New York
```

Request Sqoop without the Data Connector for Oracle and Hadoop import this data using a system located in Melbourne Australia. The timestamps are imported correctly but the local time zone has to be guessed. If multiple systems in different locale were executing the Sqoop import it would be very difficult to diagnose the cause of the data corruption.

```
2010-04-04 02:59:00.0
2010-04-04 16:59:00.0
```

Sqoop with the Data Connector for Oracle and Hadoop explicitly states the time zone portion of the data imported into Hadoop. The local time zone is GMT by default. You can set the local time zone with parameter:

```
-Doracle.sessionTimeZone=Australia/Melbourne
```

The Data Connector for Oracle and Hadoop would import these two timestamps as:

```
2010-04-04 02:59:00.0 Australia/Melbourne
2010-04-04 16:59:00.0 Australia/Melbourne
```

25.8.6.5. java.sql.Timestamp

To use Sqoop's handling of date and timestamp data types when importing data from Oracle use the following parameter:

```
-Doraoop.timestamp.string=false
```

	Note
Sqoop's handling of date and timestamp data types does not store the timezone. However, some developers may prefer Sqoop's handling as the Data Connector for Oracle and Hadoop converts date and timestamp data types to string. This may not work for some developers as the string will require parsing later in the workflow.	

25.8.6.6. Export Date And Timestamp Data Types into Oracle

Ensure the data in the HDFS file fits the required format exactly before using Sqoop to export the data into Oracle.

	Note
The Sqoop export command will fail if the data is not in the required format.ff = Fractional secondTZR = Time Zone Region	

Oracle Data Type	Required Format of The Data in the HDFS File
DATE	yyyy-mm-dd hh24:mi:ss
TIMESTAMP	yyyy-mm-dd hh24:mi:ss.ff

Oracle Data Type	Required Format of The Data in the HDFS File
TIMESTAMPTZ	yyyy-mm-dd hh24:mi:ss.ff TZR
TIMESTAMPLTZ	yyyy-mm-dd hh24:mi:ss.ff TZR

25.8.7. Configure The Data Connector for Oracle and Hadoop

25.8.7.1. oraoop-site-template.xml

The oraoop-site-template.xml file is supplied with the Data Connector for Oracle and Hadoop. It contains a number of ALTER SESSION statements that are used to initialize the Oracle sessions created by the Data Connector for Oracle and Hadoop.

If you need to customize these initializations to your environment then:

1. Find `oraoop-site-template.xml` in the Sqoop configuration directory.
2. Copy `oraoop-site-template.xml` to `oraoop-site.xml`.
3. Edit the `ALTER SESSION` statements in `oraoop-site.xml`.

25.8.7.2. oraoop.oracle.session.initialization.statements

The value of this property is a semicolon-delimited list of Oracle SQL statements. These statements are executed, in order, for each Oracle session created by the Data Connector for Oracle and Hadoop.

The default statements include: alter session set time_zone = '{oracle.sessionTimeZone|GMT}';

This statement initializes the timezone of the JDBC client. This ensures that data from columns of type TIMESTAMP WITH LOCAL TIMEZONE are correctly adjusted into the timezone of the client and not kept in the timezone of the Oracle database.

	Note
There is an explanation to the text within the curly-braces. See “Expressions in oraoop-site.xml” for more information..A list of the time zones supported by your Oracle database is available by executing the following query: SELECT TZNAME FROM V$TIMEZONE_NAMES;	

alter session disable parallel query;

This statement instructs Oracle to not parallelize SQL statements executed by the Data Connector for Oracle and Hadoop sessions. This Oracle feature is disabled because the Map/Reduce job launched by Sqoop is the mechanism used for parallelization.

It is recommended that you not enable parallel query because it can have an adverse effect the load on the Oracle instance and on the balance between the Data Connector for Oracle and Hadoop mappers.

Some export operations are performed in parallel where deemed appropriate by the Data Connector for Oracle and Hadoop. See “Parallelization” for more information. alter session set "_serial_direct_read"=true; This statement instructs Oracle to bypass the buffer cache. This is used to prevent Oracle from filling its buffers with the data being read by the Data Connector for Oracle and Hadoop, therefore

diminishing its capacity to cache higher prioritized data. Hence, this statement is intended to minimize the Data Connector for Oracle and Hadoop's impact on the immediate future performance of the Oracle database. --alter session set events '10046 trace name context forever, level 8'; This statement has been commented-out. To allow tracing, remove the comment token "–" from the start of the line.

	Note
These statements are placed on separate lines for readability. They do not need to be placed on separate lines.A statement can be commented-out via the standard Oracle double-hyphen token: "–". The comment takes effect until the next semicolon.	

25.8.7.3. oraoop.table.import.where.clause.location

SUBSPLIT（默认）

设置为该值时，where子句将应用于每个子查询，这些子查询用于从Oracle表中检索数据。

一个Sqoop命令，例如：

```
sqoop import -D oraoop.table.import.where.clause.location=SUBSPLIT --table JUNK --where "owner like 'G%'"
```

生成以下形式的SQL查询：

```
SELECT OWNER，OBJECT_NAME
  从垃圾
 在哪里（（rowid> =
            dbms_rowid.rowid_create（1，113320，1024，4223664，0）
          AND rowid <=
              dbms_rowid.rowid_create（1，113320，1024，4223671，32767）））
   AND（所有者，例如“ G%”）
```

```
全联盟
SELECT OWNER，OBJECT_NAME
  从垃圾
 在哪里（（rowid> =
             dbms_rowid.rowid_create（1，113320，1024，4223672，0）
          AND rowid <=
             dbms_rowid.rowid_create（1，113320，1024，4223679，32767）））
   AND（所有者，例如“ G%”）
```

分裂

设置为该值时，where子句将应用于每个拆分/映射器使用的整个SQL语句。

一个Sqoop命令，例如：

```
sqoop import -D oraoop.table.import.where.clause.location=SPLIT --table JUNK --where "rownum ⇐ 10"
```

生成以下形式的SQL查询：

```
SELECT OWNER，OBJECT_NAME
  从 （
        SELECT OWNER，OBJECT_NAME
          从垃圾
         在哪里（（rowid> =
                    dbms_rowid.rowid_create（1，113320，1024，4223664，0）
                 AND rowid <=
                    dbms_rowid.rowid_create（1，113320，1024，4223671，32767）））
        全联盟
        SELECT OWNER，OBJECT_NAME
          从垃圾
         在哪里（（rowid> =
                    dbms_rowid.rowid_create（1，113320，1024，4223672，0）
                 AND rowid <=
                    dbms_rowid.rowid_create（1，113320，1024，4223679,32767）））
       ）
 其中rownum <= 10
```

	笔记
在此示例中，每个映射器最多导入10行。SPLIT子句可能比SUBSPLIT子句导致更大的开销，因为在将数据流传输到映射器之前，需要完全实现UNION语句。但	

是，如果要限制每个映射器处理的总行数，则可能希望使用SPLIT。

25.8.7.4。oracle.row.fetch.size

此属性的值是一个整数，用于指定Oracle JDBC驱动程序在每次网络到数据库的往返中应获取的行数。默认值为5000。

如果更改此设置，则在“ Map-Reduce”作业期间，更改的确认会显示在映射器的日志中。

25.8.7.5。oraoop.import.hint

将Oracle优化程序提示添加到IMPORT作业的SELECT语句中，如下所示：

```
SELECT / * + NO_INDEX（t）* / *来自员工；
```

默认提示是 NO_INDEX(t)

	笔记
提示可以添加到命令行。有关更多信息，请参见“从Oracle导入数据”。有关Oracle优化器提示的更多信息，请参见《 Oracle数据库性能调优指南》（使用优化器提示）。要关闭提示，请在<value>元素之间插入一个空格。 <属性> <name> oraoop.import.hint </ name> <value> </ value> </ property>	

25.8.7.6。oraoop.oracle.append.values.hint.usage

此属性的值为以下之一：AUTO / ON / OFF。汽车

AUTO是默认值。

当前，AUTO等效于OFF。上在导出期间，Oracle和Hadoop的数据连接器使用直接路径写入来绕过缓冲区高速缓存，以填充目标Oracle表。Oracle仅允许单个会话随时对特定表执行直接写操作，因此具有序列化对表的写操作的效果。这可能会降低吞吐量，特别是如果映射器的数量很高时。但是，对于DBWR非常繁忙的数据库，或到基础表的IO带宽很窄的数据库（例如，表驻留在单个磁盘主轴上），则设置 oraoop.oracle.append.values.hint.usage 为ON可能会减少Oracle数据库的负载并可能增加吞吐量。离开在导出期间，用于Oracle和Hadoop的数据连接器不使用 APPEND_VALUES Oracle提示。

	笔记
此参数仅对Oracle 11g第2版及更高版本有效。	

25.8.7.7。mapred.map.tasks.speculative.execution

默认情况下，针对Oracle和Hadoop的数据连接器禁用推测性执行。这样可以避免在Oracle数据库上增加冗余负载。

如果启用了推测执行，则Hadoop可能会启动多个映射器以读取相同的数据块，从而增加了数据库的总体负载。

25.8.7.8。oraoop.block.allocation

此设置确定如何将Oracle的数据块分配给Map-Reduce映射器。

	笔记

适用于进口。不适用于出口。

ROUNDROBIN（默认）

Oracle块的每个块都以轮询方式分配给映射器。这有助于防止从Oracle数据文件开始就为映射器之一分配很大比例的通常较小的块。这样做还有助于防止从Oracle数据文件的末尾为其他映射器中的一个分配较大比例的通常较大的块。

使用此方法有助于确保为所有映射器分配相似的工作量。随机的Oracle块列表是随机的，然后通过循环方法分配给映射器。这样做的好处是，在任何给定的时间点，每个映射器都从不同的Oracle数据文件中读取数据的机会增加了。如果Oracle数据文件位于不同的主轴上，则这将增加总体IO吞吐量。顺序的

Oracle块的每个块都按顺序分配给映射器。这会导致每个映射器都倾向于顺序读取一个大的，连续的Oracle数据文件。此方法的性能不可能超过循环方法，并且更可能分配映射器之间的工作差异很大。

通常不建议使用此方法。

25.8.7.9。oraoop.import.omit.lobs.long

此设置可用于忽略要导入的Oracle表中的所有LOB列（BLOB，CLOB和NCLOB）和LONG列。这在故障排除方面很有优势，因为它提供了一种从导入中排除所有基于LOB的数据的便捷方法。

25.8.7.10。oraoop.locations

	笔记

适用于进口。不适用于出口。

By default, four mappers are used for a Sqoop import job. The number of mappers can be altered via the Sqoop `--num-mappers` parameter.

If the data-nodes in your Hadoop cluster have 4 task-slots (that is they are 4-CPU core machines) it is likely for all four mappers to execute on the same machine. Therefore, IO may be concentrated between the Oracle database and a single machine.

This setting allows you to control which DataNodes in your Hadoop cluster each mapper executes on. By assigning each mapper to a separate machine you may improve the overall IO performance for the job. This will also have the side-effect of the imported data being more diluted across the machines in the cluster. (HDFS replication will dilute the data across the cluster anyway.)

将计算机名称指定为逗号分隔的列表。位置以循环方式分配给每个映射器。

如果使用EC2，请指定计算机的内部名称。这是从Sqoop命令行使用此参数的示例：

```
$ sqoop import -D oraoop.locations=ip-10-250-23-225.ec2.internal,ip-10-250-107-32.ec2.internal,ip-10-250-207-2.ec2.internal,ip-10-250-27-114.ec2.internal --direct --connect...
```

25.8.7.11。sqoop.connection.factories

如果Oracle和Hadoop的数据连接器不能接受作业，则此设置确定行为。默认情况下，Sqoop接受Data Connector for Oracle和Hadoop拒绝的作业。

`org.apache.sqoop.manager.oracle.OraOopManagerFactory` 如果Oracle和Hadoop的数据连接器无法接受作业，则将值设置为希望作业失败的时间。

25.8.7.12。oraoop-site.xml中的表达式

大括号{和}中包含的文本是在执行SQL语句之前要评估的表达式。该表达式包含配置属性的名称（可选），后跟一个默认值（如果尚未设置该属性的话）。管道| 字符用于分隔属性名称和默认值。

例如：当执行此Sqoop命令时$ sqoop import -D oracle.sessionTimeZone=US/Hawaii --direct --connect oraoop-site.xml中的语句 alter session set time_zone ='{oracle.sessionTimeZone|GMT}'; 成为 alter session set time_zone = 'US/Hawaii' 如果未设置oracle.sessionTimeZone属性，则此语句将使用指定的默认值，并成为 alter session set time_zone = 'GMT'

	笔记
如果您希望一直使用此设置，则可以 oracle.sessionTimeZone 在 sqoop-site.xml 文件中指定该属性 。	

25.8.8。对适用于Oracle和Hadoop的数据连接器进行故障排除

25.8.8.1。引用Oracle所有者和表

如果需要对Oracle表的所有者加 引号，请使用：	$ 这等效于： “ Scott” .customers sqoop import ... --table "\"\"Scott\".customers\""

如果需要在Oracle表中加引号，请使用：	$ 这等效于： scott。“客户” sqoop import … --table "\"scott.\"Customers\"\""
如果 需要同时引用Oracle表的所有者和表本身，请使用：	$ 这等效于： “ Scott”。“ Customers” sqoop import … --table "\"\"Scott\".\"Customers\"\""

	笔记
HDFS输出目录的名称类似于：/user/username/”Scott”.”Customers”如果表名包含$字符，则可能需要在Unix shell中对其进行转义。例如，ctxsys模式中的dr $ object表将被称为：$ sqoop import … --table "ctxsys.dr\$object"	

25.8.8.2。引用Oracle列

如果需要用Oracle表的列名加引号，请使用

$ sqoop import … --table customers --columns "\"\"first name\"\""

这等效于： select "first name" from customers 如果需要引用多列，请使用

$ sqoop import … --table customers --columns "\"\"first name\",\"last name\",\"region name\"\""

这等效于： select "first name", "last name", "region name" from customers

25.8.8.3。确认用于Oracle和Hadoop的数据连接器可以初始化Oracle会话

如果Sqoop输出包括反馈如包含在下面接着的配置属性 oraoop-site-template.xml 和 oraoop-site.xml 由Hadoop的已经被加载，并且可以通过数据连接器用于Oracle和Hadoop进行访问。

```
14/07/08 15:21:13 INFO oracle.OracleConnectionFactory: Initializing Oracle session with SQL
```

25.8.8.4。检查Sqoop调试日志中的错误消息

有关Sqoop导入期间遇到的任何错误的更多信息，请参考执行导入的每个（默认为4个）映射器生成的日志文件。

可以通过Map-Reduce Job Tracker的网页获取日志。

在Sqoop用户组网站上，将这些日志文件包含在您提出的任何帮助请求中。

25.8.8.5。导出：检查表兼容

检查表，尤其是在分析错误的情况下。

- 确保HDFS文件中包含的字段与Oracle表中的列相同。如果它们不相同，则Sqoop动态生成的用于解析HDFS文件的Java代码在读取文件时将引发错误-导致导出失败。在Oracle中创建表时，请确保表模板的定义与HDFS文件的定义相同。
- 确保支持表中的数据类型。有关更多信息，请参见“支持的数据类型”。
- 是否使用基于日期和时区的数据类型？有关更多信息，请参见“将日期和时间戳记数据类型导出到Oracle”。

25.8.8.6。出口：平行化

```
-D oraoop.export.oracle.parallelization.enabled=false
```

如果看到并行化错误，则可以决定对Oracle查询禁用并行化。

25.8.8.7。导出：检查oraoop.oracle.append.values.hint.usage

如果Oracle表包含BINARY_DOUBLE或BINARY_FLOAT列，并且要导出的HDFS文件在这两种列类型中都包含NULL值，则不应将oraoop.oracle.append.values.hint.usage参数设置为ON。这样做会导致错误： `ORA-12838: cannot read/modify an object after modifying it in parallel` 。

25.8.8.8。打开详细

在Sqoop命令行上打开verbose。

```
--verbose
```

检查Sqoop stdout（标准输出）和映射器日志，以获取有关问题可能出在哪里的信息。

26.获得支持

一些常规信息可从 http://sqoop.apache.org/获得。

通过https://issues.apache.org/jira/browse/SQOOP向问题跟踪器报告Sqoop中的错误 。

有关Sqoop用法的问题和讨论，请直接访问 sqoop-user邮件列表。

在联系任何一个论坛之前，请运行带有该 --verbose 标志的Sqoop作业， 以获取尽可能多的调试信息。还报告返回的字符串 sqoop version 以及您正在运行的Hadoop的版本（ hadoop version ）。

27.故障排除

27.1。常规故障排除过程27.2。特定的故障排除技巧27.2.1。Oracle：连接重置错误27.2.2。Oracle：区分大小写的目录查询错误27.2.3。MySQL：连接失败27.2.4。Oracle：ORA-00933错误（SQL命令未正确结束）27.2.5。MySQL：从MySQL导入TINYINT（1）的行为异常

27.1。常规故障排除过程

应该按照以下步骤对运行Sqoop时遇到的任何故障进行故障排除。

- 通过再次执行同一命令并指定 --verbose 选项来打开详细输出。这样会在控制台上产生更多的调试输出，可以对其进行检查以识别任何明显的错误。
- 查看来自Hadoop的任务日志，以查看其中是否记录了任何特定的故障。任务执行过程中发生的故障可能未正确中继到控制台。
- 确保存在必要的输入文件或输入/输出表，并且Sqoop执行身份或连接到数据库的用户可以访问这些文件。可能存在必要的文件或表，但Sqoop连接的特定用户没有访问这些文件的必要权限。
- 如果要执行复合操作（例如填充Hive表或分区），请尝试将作业分为两个单独的操作，以查看问题的真正出处。例如，如果创建并填充Hive表的导入失败，则可以将其分为两个步骤-第一个步骤是单独执行导入，第二个步骤是使用该 create-hive-table 工具在不进行导入的情况下创建Hive表。虽然这不能解决填充Hive表的原始用例，但确实可以将问题缩小到常规导入或在Hive表的创建和填充期间。

在邮件列表档案和JIRA中搜索与该问题有关的关键字。您可能会找到在那里讨论的解决方案，可以帮助您解决问题或解决问题。

27.2。特定的故障排除技巧

27.2.1。Oracle：连接重置错误

问题：将默认的Sqoop连接器用于Oracle时，确实会传输一些数据，但是在map-reduce作业期间，报告了许多错误，如下所示：

```
11/05/26 16:23:47 INFO mapred.JobClient：任务ID：try_201105261333_0002_m_000002_0，状态：FAILED
java.lang.RuntimeException：java.lang.RuntimeException：java.sql.SQLRecoverableException：IO错误：连接重置
在com.cloudera.sqoop.mapreduce.db.DBInputFormat.setConf（DBInputFormat.java:164）
在org.apache.hadoop.util.ReflectionUtils.setConf（ReflectionUtils.java:62）
在org.apache.hadoop.util.ReflectionUtils.newInstance（ReflectionUtils.java:117）
在org.apache.hadoop.mapred.MapTask.runNewMapper（MapTask.java:605）
在org.apache.hadoop.mapred.MapTask.run（MapTask.java:322）
在org.apache.hadoop.mapred.Child $ 4.run（Child.java:268）
在java.security.AccessController.doPrivileged（本机方法）
在javax.security.auth.Subject.doAs（Subject.java:396）
在org.apache.hadoop.security.UserGroupInformation.doAs（UserGroupInformation.java:1115）
在org.apache.hadoop.mapred.Child.main（Child.java:262）
引起原因：java.lang.RuntimeException：java.sql.SQLRecoverableException：IO错误：连接重置
在com.cloudera.sqoop.mapreduce.db.DBInputFormat.getConnection（DBInputFormat.java:190）
在com.cloudera.sqoop.mapreduce.db.DBInputFormat.setConf（DBInputFormat.java:159）
...另外9个
引起原因：java.sql.SQLRecoverableException：IO错误：连接重置
在oracle.jdbc.driver.T4CConnection.logon（T4CConnection.java:428）
在oracle.jdbc.driver.PhysicalConnection。<init>（PhysicalConnection.java:536）上
在oracle.jdbc.driver.T4CConnection。<init>（T4CConnection.java:228）
在oracle.jdbc.driver.T4CDriverExtension.getConnection（T4CDriverExtension.java:32）
在oracle.jdbc.driver.OracleDriver.connect（OracleDriver.java:521）
在java.sql.DriverManager.getConnection（DriverManager.java:582）
在java.sql.DriverManager.getConnection（DriverManager.java:185）
在com.cloudera.sqoop.mapreduce.db.DBConfiguration.getConnection（DBConfiguration.java:152）
```

```
在com.cloudera.sqoop.mapreduce.db.DBInputFormat.getConnection（DBInputFormat.java:184）
...另外10个
引起原因：java.net.SocketException：连接重置
在java.net.SocketOutputStream.socketWrite（SocketOutputStream.java:96）
在java.net.SocketOutputStream.write（SocketOutputStream.java:136）
在oracle.net.ns.DataPacket.send（DataPacket.java:199）
在oracle.net.ns.NetOutputStream.flush（NetOutputStream.java:211）
在oracle.net.ns.NetInputStream.getNextPacket（NetInputStream.java:227）
在oracle.net.ns.NetInputStream.read（NetInputStream.java:175）
在oracle.net.ns.NetInputStream.read（NetInputStream.java:100）
在oracle.net.ns.NetInputStream.read（NetInputStream.java:85）
在oracle.jdbc.driver.T4CSocketInputStreamWrapper.readNextPacket（T4CSocketInputStreamWrapper.java:123）
在oracle.jdbc.driver.T4CSocketInputStreamWrapper.read（T4CSocketInputStreamWrapper.java:79）
在oracle.jdbc.driver.T4CMAREngine.unmarshalUB1（T4CMAREngine.java:1122）
在oracle.jdbc.driver.T4CMAREngine.unmarshalSB1（T4CMAREngine.java:1099）
在oracle.jdbc.driver.T4CTTIfun.receive（T4CTTIfun.java:288）
在oracle.jdbc.driver.T4CTTIfun.doRPC（T4CTTIfun.java:191）
在oracle.jdbc.driver.T4CTTIoauthenticate.doOAUTH（T4CTTIoauthenticate.java:366）
在oracle.jdbc.driver.T4CTTIoauthenticate.doOAUTH（T4CTTIoauthenticate.java:752）
在oracle.jdbc.driver.T4CConnection.logon（T4CConnection.java:366）
...另外18个
```

解决方案：出现此问题的主要原因是执行映射任务的主机上缺少快速随机数生成设备。在典型的Linux系统上，可以通过在 java.security 文件中设置以下属性来解决此问题：

```
securerandom.source =文件：/ dev /../ dev / urandom
```

该 java.security 文件可以在 $JAVA_HOME/jre/lib/security 目录下找到。另外，也可以通过以下命令在命令行上指定此属性：

```
-D mapred.child.java.opts =“-Djava.security.egd = file：/ dev /../ dev / urandom”
```

请注意，指定此怪异的路径非常重要， /dev/../dev/urandom 因为它是Java错误 6202721引起的，否则 /dev/urandom 将被忽略并替换为 /dev/random 。

27.2.2。Oracle：区分大小写的目录查询错误

问题：使用Oracle时，当Sqoop无法找出列名时，您可能会遇到问题。发生这种情况是因为Sqoop用于Oracle的目录查询期望为用户名和表名指定正确的大小写。

一个使用–hive-import并导致NullPointerException的示例：

```
1/09/21 17:18:49 INFO管理器.OracleManager：时区已设置为
格林威治标准时间
11/09/21 17:18:49 DEBUG manager.SqlManager：将fetchSize用于下一个
查询：1000
11/09/21 17:18:49 INFO manager.SqlManager：执行SQL语句：
SELECT t。* FROM addlabel_pris t WHERE 1 = 0
11/09/21 17:18:49 DEBUG管理器.OracleManager $ ConnCache：缓存
释放了jdbc：oracle：thin的连接：
11/09/21 17:18:49 ERROR sqoop.Sqoop：运行Sqoop时出现异常：
java.lang.NullPointerException
java.lang.NullPointerException
在com.cloudera.sqoop.hive.TableDefWriter.getCreateTableStmt（TableDefWriter.java:148）
在com.cloudera.sqoop.hive.HiveImport.importTable（HiveImport.java:187）
在com.cloudera.sqoop.tool.ImportTool.importTable（ImportTool.java:362）
在com.cloudera.sqoop.tool.ImportTool.run（ImportTool.java:423）
在com.cloudera.sqoop.Sqoop.run（Sqoop.java:144）
在org.apache.hadoop.util.ToolRunner.run（ToolRunner.java:65）
在com.cloudera.sqoop.Sqoop.runSqoop（Sqoop.java:180）
在com.cloudera.sqoop.Sqoop.runTool（Sqoop.java:219）
在com.cloudera.sqoop.Sqoop.runTool（Sqoop.java:228）
在com.cloudera.sqoop.Sqoop.main（Sqoop.java:237）
```

解决方案：

1. 以大写形式指定Sqoop所连接的用户名（除非它是用引号内的大小写混合的）创建的。
2. 用大写字母指定要使用的表名（除非它是用引号内的大小写混合/小写创建的）。

27.2.3。MySQL：连接失败

问题：在将MySQL表导入Sqoop时，如果您没有通过网络访问MySQL数据库所必需的权限，则可能会出现以下连接失败。

```
引起原因：com.mysql.jdbc.exceptions.jdbc4.CommunicationsException：通信链接失败
```

解决方案：首先，确认您可以从运行Sqoop的节点连接到数据库：

```
$ mysql --host = <IP地址> --database = test --user = <用户名> --password = <密码>
```

如果可行，则排除客户端网络配置或安全性/身份验证配置中的任何问题。

将服务器的网络端口添加到my.cnf文件中 /etc/my.cnf ：

```
[mysqld]
端口= xxxx
```

设置用户帐户以通过Sqoop连接。向用户授予访问网络上数据库的权限：（1.）以root用户身份登录MySQL mysql -u root -p<ThisIsMyPassword> 。（2.）发出以下命令：

```
mysql>授予test。*的所有特权给'testpassword'标识的'testuser'@'%'
```

注意，这样做将使测试用户能够从任何IP地址连接到MySQL服务器。尽管这将起作用，但不建议在生产环境中使用。我们建议您咨询DBA，以根据设置拓扑授予必要的特权。

如果数据库服务器的IP地址发生更改，除非将其绑定到服务器中的静态主机名，否则传递给Sqoop的连接字符串也将需要更改。

27.2.4。Oracle：ORA-00933错误（SQL命令未正确结束）

问题：使用Oracle时，当Sqoop命令显式指定–driver <driver name>选项时，您可能会遇到以下问题。当Sqoop命令中包含驱动程序选项时，内置连接管理器选择默认为通用连接管理器，这会导致Oracle出现此问题。如果未指定驱动程序选项，则内置的连接管理器选择机制将选择Oracle特定的连接管理器，该管理器为Oracle生成有效的SQL，并使用驱动程序“ oracle.jdbc.OracleDriver”。

```
错误manager.SqlManager：执行语句出错：
java.sql.SQLSyntaxErrorException：ORA-00933：SQL命令未正确结束
```

解决方案：忽略–driver oracle.jdbc.driver.OracleDriver选项，然后重新运行Sqoop命令。

27.2.5。MySQL：从MySQL导入TINYINT（1）的行为异常

问题：Sqoop将TINYINT（1）列视为布尔值，例如这导致HIVE导入问题。这是因为默认情况下，MySQL JDBC连接器将TINYINT（1）映射到java.sql.Types.BIT，Sqoop默认将其映射到布尔值。

解决方案：一个更干净的解决方案是通过添加 tinyInt1isBit=false 到JDBC路径（创建类似 jdbc:mysql://localhost/test?tinyInt1isBit=false ）来强制MySQL JDBC连接器停止将TINYINT（1）转换为java.sql.Types.BIT 。另一个解决方案是显式覆盖数据类型TINYINT（1）列的列映射。例如，如果列名称为foo，则在导入期间将以下选项传递给Sqoop：–map-column-hive foo = tinyint。对于非Hive导入HDFS的情况，请使用–map-column-java foo = integer。
本文档是从Sqoop源构建的，可从 https://git-wip-us.apache.org/repos/asf?p=sqoop.git获取。

其他关联文章:

1. Sqoop **导入实战(一)**
2. Sqoop **导入实战(二)**

delucia / 2021-05-10 / Sqoop