Sqoop

Data Imports

Data Exports

Integration with Hadoop Ecosystem

SQOOP…

- Overview
- Why Sqoop so popular
- How to get it working
- Work-flow
- Rdbms-Hdfs-Hive

SQOOP…

Prerequisite

1. Mysql db which is available and db user has all privilege rights
2. The db should have at least one table with few rows in it – preferable close to 100 to be able to see the no of mappers kicking in
3. Internet access on system or Keep the sqoop tar downloaded from apache location

MySQL - Installation

http://dev.mysql.com/downloads/mysql/

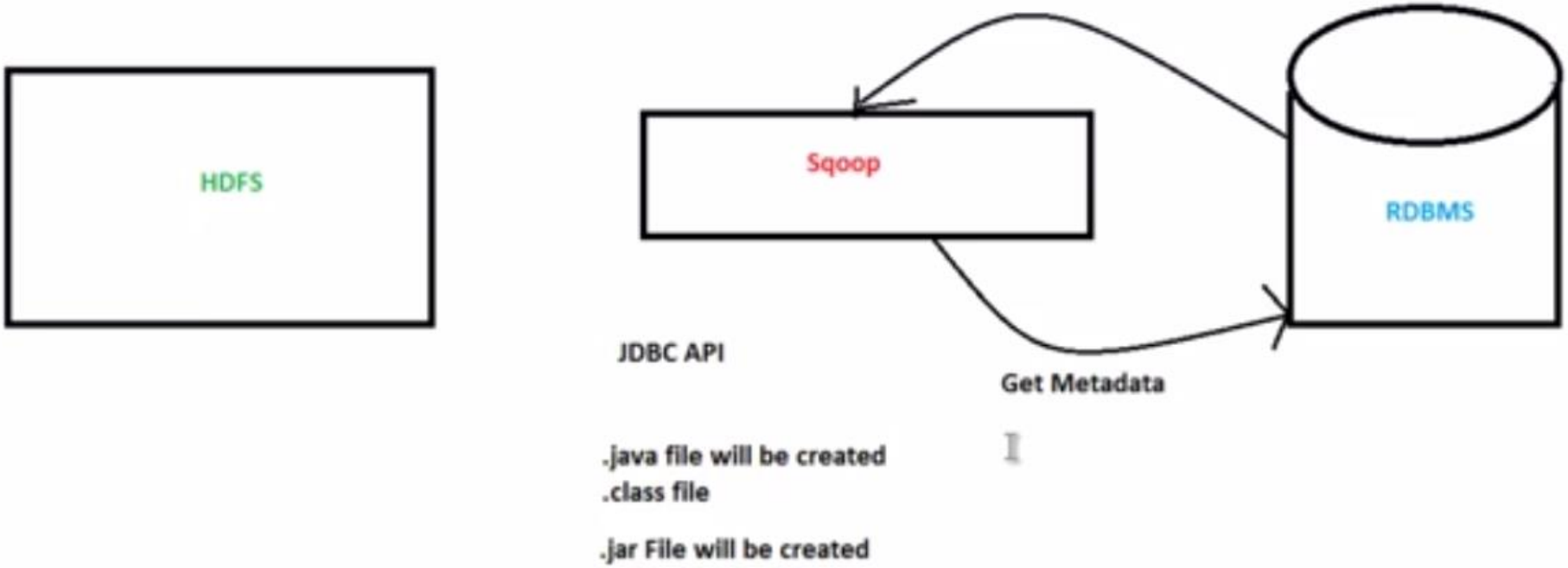
32 Bit OR 64 bit based on Windows version can be downloaded from here

You will have an MSI file downloaded .

SQOOP…

Sqoop is used widely in industry as it is going to be your first apache product from the minute you have decided to move from relational DB to Hadoop ecosystem

Working



Working in Step By Step:

Step 1: Sqoop send the request to Relational DB to send the return the meta data information about the table(Metadata here is the data about the table in relational DB)

Step 2: From the received information it will generate the java classes (Reason why you should have java configured before get it working-Sqoop internally uses **jdbc** API to generate data).

Step 3:Now sqoop (As its written in java –tries to package the compiled classes to be able to generate table structure) , post compiling creates jar file(Java packaging standard)

Lets Dive in:

Say you had a table like

EmpID	FirstName	LastName	Manager
1	George	Dsouza	NULL
2	Jackie	Parera	1
3	Tejas	Shah	1
4	Kunal	Vyas	1
5	Hiren	Desai	4
6	Mohammed	Hanif	5
7	Sajid	Pathan	2
8	Elaine	Benis	7
9	Vishal	Manghnani	3
10	Manoj	Diwakaran	6

Now you have decided that you do not like relational structure any more as you expect this table to grow really-really large hence you want this to get moved to Hadoop ecosystem and get rid of licensing

PS: sqoop needs to have primary key t work best – but no worries if your table structure inherently doens't have it , it will create that for you but without affecting your table metadata structure

SQOOP…

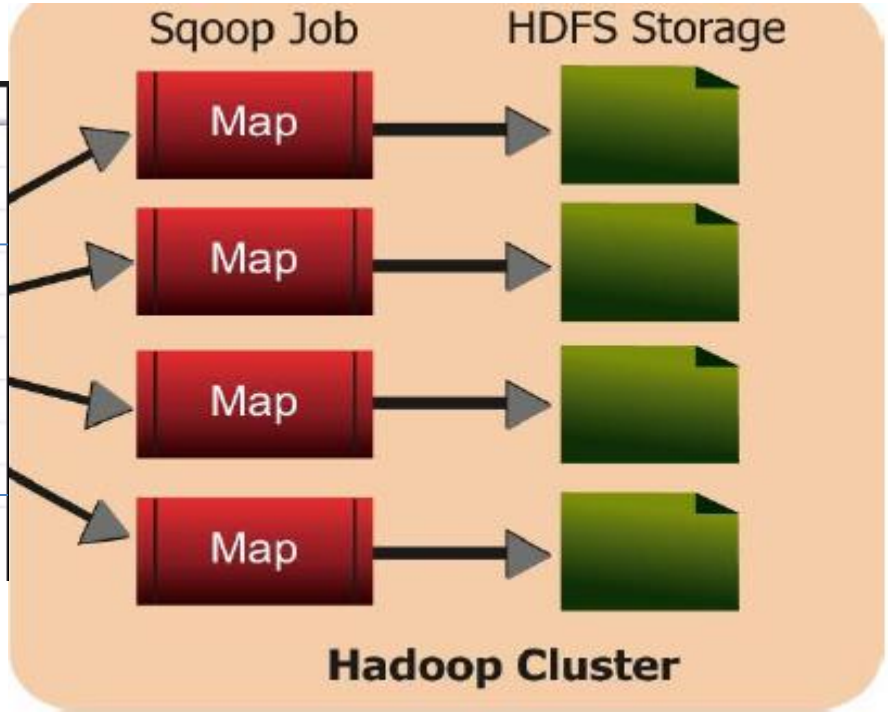
Internal Workflow:

1. Sqoop asks fro meta data information from Relation DB
2. Relational DB returns the required request
3. Based on meta data information sqoop generates java classes
4. Based on **primary id partitioning happens in table** as multiple mappers will importing data as the same time

Lets understand this line in more detail as this is crux of why sqoop works faster than any other importing technique and how come mapper comes in to picture

SQOOP…

	EmpID	FirstName	LastName	Manager
1	1	George	Dsouza	NULL
	2	Jackie	Parera	1
	3	Tejas	Shah	1
2	4	Kunal	Vyas	1
	5	Hiren	Desai	4
	6	Mohammed	Hanif	5
3	7	Sajid	Pathan	2
	8	Elaine	Benis	7
	9	Vishal	Manghnani	3
4	10	Manoj	Diwakaran	6



This image is symbolic and of course you do not need 4 mappers to just import a row size of 10 and sqoop is intelligent enough to decide on optimum number of mappers required for the job

SQOOP…

- Lets Understand it in terms of what would have happened technically
- Sqoop fires and import request
- Internally what it needs to know is what is min and max primary key value is in there
- Select min(emp_id) as min_value ,max(emp_id) as max_value from employee
- And then for each mapper
- Select * from employee where emp_id **between** min_value and max_value(Scope of each mapper)

Example:

Where relation DB is present

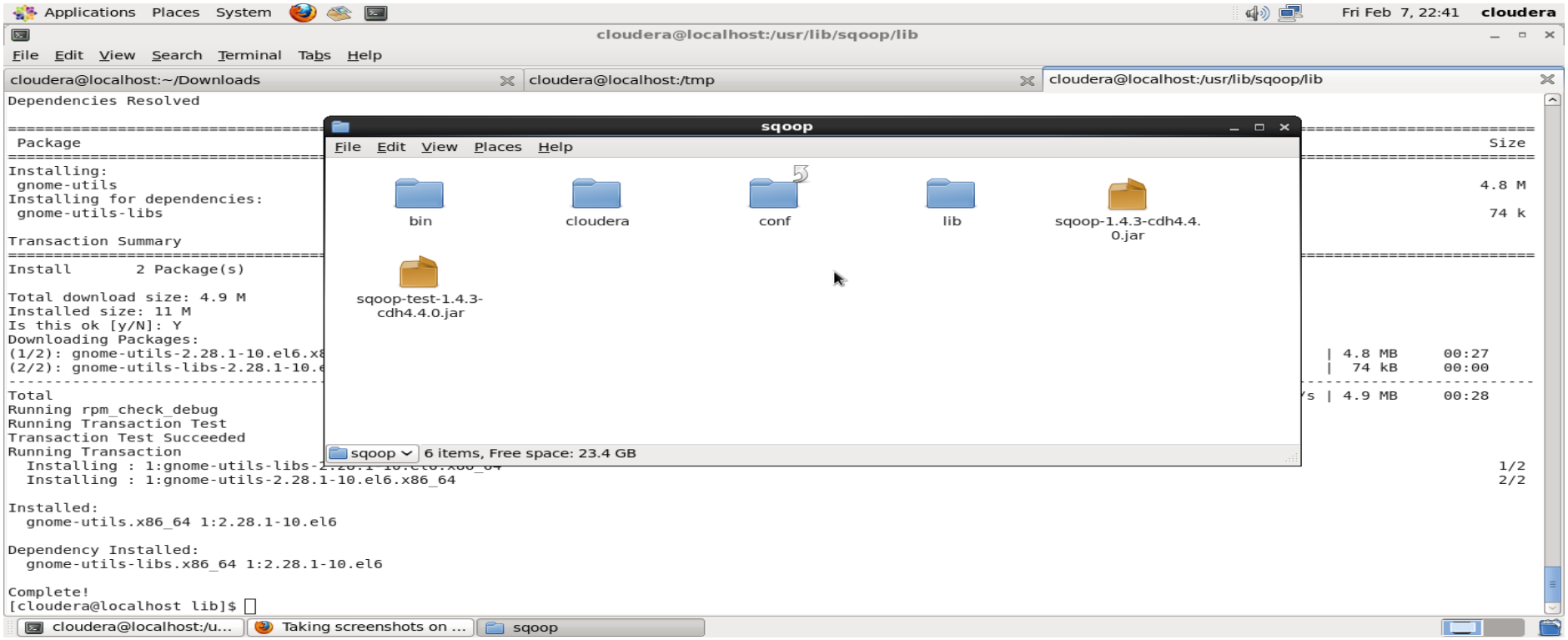
sqoop import \
--**connect** jdbc:mysql://<**ip address**>\<database name>
--table <mysql_table name>
--username <username_for_mysql_user> --password <Password>
-m <**number of mappers to run**>
--target-dir <target directory where we need copied data>

If you do not want sqoop to decide on your behalf

Double "–" because its subcommand of import

Now how to setup Sqoop and see it working

- Write sqoop.apache.org on google and you should be able to get it
- Download and extract it at a convenient location This is how its going to look like post extraction



SQOOP…

Write sqoop on the terminal after adding the sqoop to PATH variable

Your system might complain for HADOOP_COMMON_HOME,HADOOP_MAPRED_HOME ,HIVE_HOME,HBASE_HOME,HCAT_HOME . You could set all of it at once in sqoop-env-template.sh

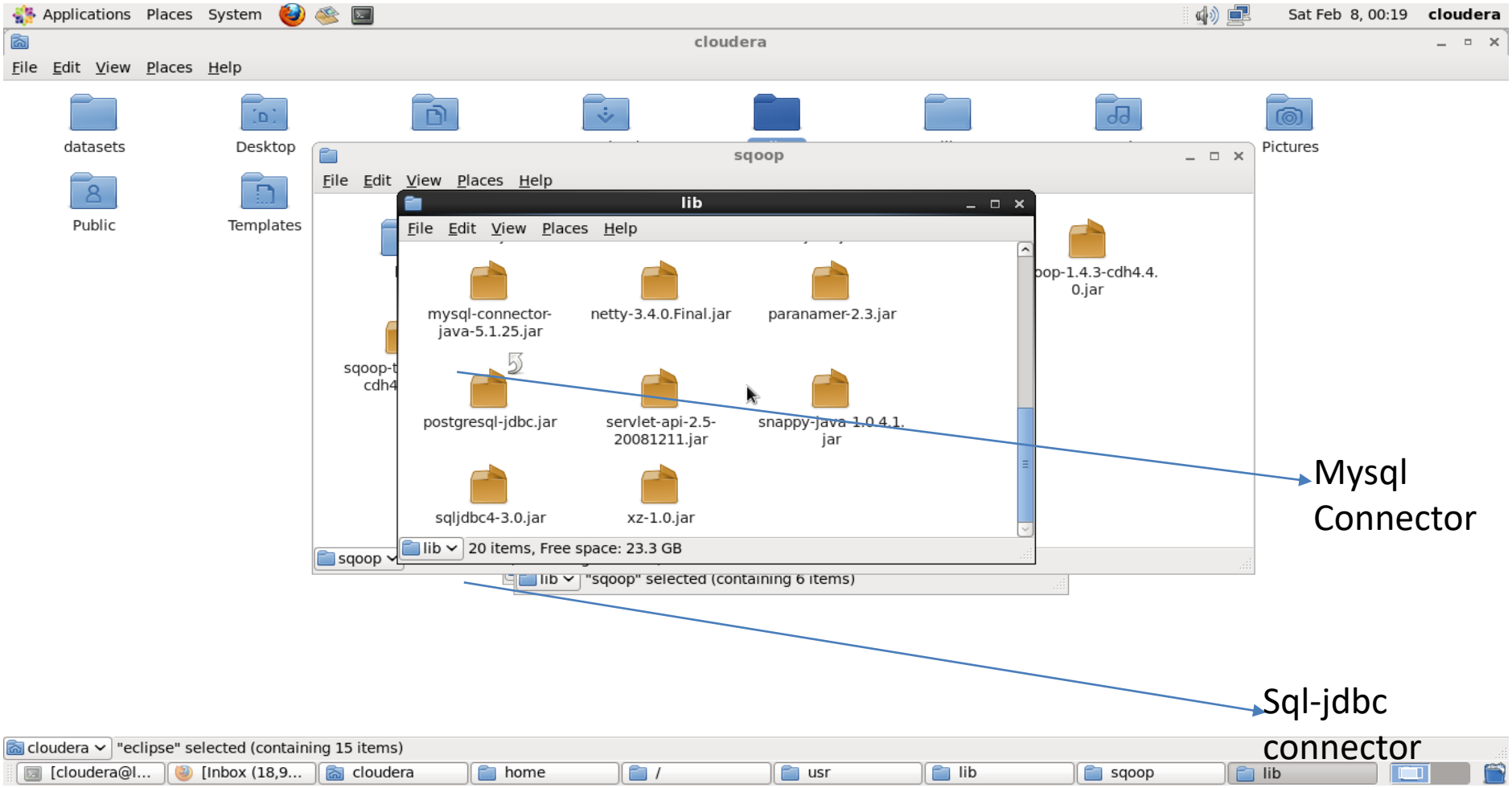
You will find this file in conf folder else search for this by firing command

Find –iname sqoop-env*.sh (On Ubuntu-lnux)

Download connector for your DB (mysql , oracle,Teradata,DB2 etc) and place it in lib folder

Also copy sqljdbc**x**.jar **(replace x with version no)** in lib as well

Typically your sqoop lib folder will have these



Sqoop

- An Open Source Tool used for Data Interaction between traditional RDBMs and Hadoop Environment.
- Using Sqoop, Data can be moved into HDFS/hive/hbase from MySQL/ PostgreSQL/Oracle/SQL Server/DB2 and vive versa.
- This import of data takes into distributed processing power of hadoop making it faster

Starting Sqoop

- Sqoop is a command line tool with following structure

  sqoop TOOL PROPERTY_ARGS SQOOP_ARGS [-- EXTRA_ARGS]

- TOOL indicates the operation eg: “import”, “export”.
- PROPERTY_ARGS are Java properties in the format “-Dname=value”
- SQOOP_ARGS mention various Sqoop parameters
- EXTRA_ARGS are for specialized connectors, separated from the SQOOP_ARGS with a “--”

E.g.:

- % sqoop import --connect jdbc:mysql://localhost/hadoopguide --table widgets -m 1

Starting Sqoop

- Type “sqoop help” to get all the tools available:

```
[training@localhost ~]$ sqoop help
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  merge              Merge results of incremental imports
  metastore          Run a standalone Sqoop metastore
  version            Display version information

See 'sqoop help COMMAND' for information on a specific command.
[training@localhost ~]$
```

Starting Sqoop

- For information on specific Sqoop tool, type “sqoop help TOOL”

```
[training@localhost ~]$ sqoop help import
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
   --connect <jdbc-uri>                         Specify JDBC connect
                                                string
   --connection-manager <class-name>            Specify connection manager
                                                class name
   --connection-param-file <properties-file>    Specify connection
                                                parameters file
   --driver <class-name>                        Manually specify JDBC
                                                driver class to use
   --hadoop-home <dir>                          Override $HADOOP_HOME
   --help                                       Print usage instructions
-P                                              Read password from console
   --password <password>                        Set authentication
                                                password
   --username <username>                        Set authentication
                                                username
   --verbose                                    Print more information
                                                while working

Import control arguments:
```

- Lets start by exploring import tool using SQL databases.

MySQL

- We have MySql installed in our Vm image and certain tables already present in various databases. First we will take a look at these data sets and finally play with various data sets

Logging into MySQL

$ mysql -u training -p

mysql> show databases;

```
[training@localhost ~]$ mysql -u training -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.1.66 Source distribution

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hue                |
| metastore          |
| mysql              |
| test               |
| training           |
+--------------------+
6 rows in set (0.02 sec)

mysql> 
```

Listing of tables

mysql> use training;

Mysql> show tables;

```
mysql>
mysql> use training;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_training |
+--------------------+
| Movies             |
| bible_freq         |
| cityByCountry      |
| countries          |
| shake_freq         |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

Listing of table contents

mysql> select * from table_name limit 10

```
mysql> select * from countries limit 10;
+----+---------------------+------+
| id | name                | code |
+----+---------------------+------+
|  1 | AFGHANISTAN         | AF   |
|  2 | ALBANIA             | AL   |
|  3 | ALGERIA             | DZ   |
|  4 | AMERICAN SAMOA      | AS   |
|  5 | ANDORRA             | AD   |
|  6 | ANGOLA              | AO   |
|  7 | ANGUILLA            | AI   |
|  8 | ANTARCTICA          | AQ   |
|  9 | ANTIGUA AND BARBUDA | AG   |
| 10 | ARGENTINA           | AR   |
+----+---------------------+------+
10 rows in set (0.00 sec)
```

- Now lets import this complete table into our HDFS and view its contents

Tools

Lets list all databases present on a mysql server using a “list-databases” tool.

(Note : for more information on list-databases, type “$ sqoop help list-databases”)

$ sqoop list-databases --connect “jdbc:mysql://localhost” --username training --password training

```
[training@localhost ~]$ sqoop list-databases --connect "jdbc:mysql://localhost" --username training --password training
14/04/03 15:37:31 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
14/04/03 15:37:31 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
information_schema
hue
metastore
mysql
test
training
[training@localhost ~]$
```

Similarly for listing tables,

$ sqoop list-tables --connect “jdbc:mysql://localhost/training” --username training -P

```
[training@localhost ~]$ sqoop list-tables --connect "jdbc:mysql://localhost/training" --username training -P
Enter password:
14/04/03 15:46:15 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Movies
bible_freq
cityByCountry
countries
shake_freq
[training@localhost ~]$
```

(Note: Another safe method of mentioning password will be discussed later)

Tools

“import-all-tables” imports all the tables present in the database mentioned.

$ sqoop import-all-tables --connect "jdbc:mysql://localhost/training" --username training -P -m 1

Here -m 1 specifies one mapper for each table. All the tables are downloaded in default directory.

```
[training@localhost ~]$ sqoop import-all-tables --connect "jdbc:mysql://localhost/training" --username training -P -m 1
Enter password:
14/04/03 17:23:46 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/04/03 17:23:46 INFO tool.CodeGenTool: Beginning code generation
14/04/03 17:23:46 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Movies` AS t LIMIT 1
14/04/03 17:23:46 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Movies` AS t LIMIT 1
14/04/03 17:23:46 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/45f714c70ac373d3e0be17eb4a25476a/Movies.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/04/03 17:23:47 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/45f714c70ac373d3e0be17eb4a25476a/Movies.jar
14/04/03 17:23:47 WARN manager.MySQLManager: It looks like you are importing from mysql.
```

Tools

- JobTracker Status

Cluster Summary (Heap Size is 15.56 MB/966.69 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots
0	0	10	1	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201404031601_0008	NORMAL	training	bible_freq.jar	0.00%	1	0	0.00%	0	0	NA	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201404031601_0001	NORMAL	training	countries.jar	100.00%	4	4	100.00%	0	0	NA	NA
job_201404031601_0003	NORMAL	training	cityByCountry.jar	100.00%	4	4	100.00%	0	0	NA	NA
job_201404031601_0004	NORMAL	training	cityByCountry.jar	100.00%	1	1	100.00%	0	0	NA	NA
job_201404031601_0005	NORMAL	training	countries.jar	100.00%	4	4	100.00%	0	0	NA	NA
job_201404031601_0006	NORMAL	training	countries.jar	100.00%	4	4	100.00%	0	0	NA	NA
job_201404031601_0007	NORMAL	training	Movies.jar	100.00%	1	1	100.00%	0	0	NA	NA

Tools

- Namenode

Goto : /user/training go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
MR	dir				2014-03-04 04:29	rwxr-xr-x	training	supergroup
Movies	dir				2014-04-03 17:24	rwxr-xr-x	training	supergroup
bible_freq	dir				2014-04-03 17:24	rwxr-xr-x	training	supergroup
cityByCountry	dir				2014-04-03 17:25	rwxr-xr-x	training	supergroup
counters	dir				2014-03-14 04:22	rwxr-xr-x	training	supergroup
countries	dir				2014-04-03 17:25	rwxr-xr-x	training	supergroup
out_ewc	dir				2014-03-10 02:12	rwxr-xr-x	training	supergroup
pig	dir				2014-03-13 08:17	rwxr-xr-x	training	supergroup
shake_freq	dir				2014-04-03 17:26	rwxr-xr-x	training	supergroup

Default location is /user/hadoop_user_name

Import

Importing “countries” table into our HDFS environment

$ sqoop import --connect “jdbc:mysql://localhost/training” --username training -P --table countries –target-dir /user/country_imported

```
[training@localhost ~]$ sqoop import --connect "jdbc:mysql://localhost/training" --username training -P --table countries --target-dir /user/country_imported
Enter password:
14/04/03 16:06:22 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/04/03 16:06:22 INFO tool.CodeGenTool: Beginning code generation
14/04/03 16:06:23 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `countries` AS t LIMIT 1
14/04/03 16:06:23 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `countries` AS t LIMIT 1
14/04/03 16:06:23 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/57d94434c47d04988b31551851c6ff00/countries.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/04/03 16:06:25 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/57d94434c47d04988b31551851c6ff00/countries.jar
14/04/03 16:06:25 WARN manager.MySQLManager: It looks like you are importing from mysql.
14/04/03 16:06:25 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
14/04/03 16:06:25 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
14/04/03 16:06:25 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
14/04/03 16:06:25 INFO mapreduce.ImportJobBase: Beginning import of countries
14/04/03 16:06:36 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/04/03 16:06:37 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`id`), MAX(`id`) FROM `countries`
14/04/03 16:06:38 INFO mapred.JobClient: Running job: job_201404031601_0001
14/04/03 16:06:39 INFO mapred.JobClient:  map 0% reduce 0%
14/04/03 16:07:10 INFO mapred.JobClient:  map 50% reduce 0%
```

(Note: make sure the target directory does not exists already)

Import

The default number of mappers used is 4. you can change this by appending the command by "**-m** ***number_of_mappers***"

Goto : /user/country_imported go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2014-04-03 16:07	rw-r--r--	training	supergroup
_logs	dir				2014-04-03 16:06	rwxrwxrwx	training	supergroup
part-m-00000	file	1.04 KB	1	64 MB	2014-04-03 16:07	rw-r--r--	training	supergroup
part-m-00001	file	1.1 KB	1	64 MB	2014-04-03 16:07	rw-r--r--	training	supergroup
part-m-00002	file	1.16 KB	1	64 MB	2014-04-03 16:07	rw-r--r--	training	supergroup
part-m-00003	file	1.2 KB	1	64 MB	2014-04-03 16:07	rw-r--r--	training	supergroup

Go back to DFS home

Goto : /user/country_imported go

Go back to dir listing

Advanced view/download options

```
1,AFGHANISTAN,AF
2,ALBANIA,AL
3,ALGERIA,DZ
4,AMERICAN SAMOA,AS
5,ANDORRA,AD
6,ANGOLA,AO
7,ANGUILLA,AI
8,ANTARCTICA,AQ
9,ANTIGUA AND BARBUDA,AG
10,ARGENTINA,AR
11,ARMENIA,AM
```

“where” clause

- You can place restrictions on data imported by using “where” clause.

Exercise :

- Lets import cityByCountry table where state (6th column is restricted to “Alaska”)

cityByCountry:

```
mysql> select * from cityByCountry limit 10;
+------+---------+------------------------+---------+----------+------------+
| city | country | name                   | lat     | lng      | state      |
+------+---------+------------------------+---------+----------+------------+
|   -1 |      -1 |                        | 1016.67 |  1016.67 |            |
| 6354 |     226 | Turnersville%2C%20NJ   | 39.7654 | -75.0621 | New Jersey |
|    1 |     150 | The%20Hague            | 52.0833 |      4.3 |            |
|    2 |     226 | Anchorage%2C%20AK      |   61.17 |  -150.02 | Alaska     |
|    3 |     226 | Glacier%20View%2C%20AK | 61.8167 | -147.717 | Alaska     |
|    4 |     226 | Birmingham%2C%20AL     | 33.5277 | -86.7992 | Alabama    |
|    5 |     226 | Huntsville%2C%20AL     |  34.707 | -86.6277 | Alabama    |
|    6 |     226 | Mobile%2C%20AL         | 30.6775 |  -88.089 | Alabama    |
|    7 |     226 | Montgomery%2C%20AL     | 32.3544 | -86.2843 | Alabama    |
|    8 |     226 | Tuscaloosa%2C%20AL     | 33.2377 |  -87.541 | Alabama    |
+------+---------+------------------------+---------+----------+------------+
10 rows in set (0.00 sec)

mysql> describe cityByCountry;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| city    | int(11)      | NO   | PRI | NULL    | auto_increment |
| country | int(11)      | NO   | MUL | 0       |                |
| name    | varchar(200) | NO   | MUL |         |                |
| lat     | float        | YES  |     | NULL    |                |
| lng     | float        | YES  |     | NULL    |                |
| state   | varchar(64)  | NO   |     |         |                |
+---------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

“where” clause

Solution:

```
sqoop import \
--connect “jdbc:mysql://localhost/training” \
--username training –P \
--table cityByCountry \
--target-dir /user/where_clause \
--where “state = ‘Alaska’” \
-m 1
```

“where” clause - output

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2014-04-03 16:44	rw-r--r--	training	supergroup
_logs	dir				2014-04-03 16:44	rwxrwxrwx	training	supergroup
part-m-00000	file	1.27 KB	1	64 MB	2014-04-03 16:44	rw-r--r--	training	supergroup

```
2,226,Anchorage%2C%20AK,61.17,-150.02,Alaska
3,226,Glacier%20View%2C%20AK,61.8167,-147.717,Alaska
1939,226,Fairbanks%2C%20AK,64.8371,-147.649,Alaska
5904,226,Wasilla%2C%20AK,61.5802,-149.462,Alaska
6128,226,North%20Pole%2C%20AK,64.7532,-147.356,Alaska
6941,226,Palmer%2C%20AK,61.5989,-149.11,Alaska
10129,226,SITKA%2C%20AK,57.2141,-135.447,Alaska
10230,226,HOMER%2C%20AK,59.6355,-151.522,Alaska
10297,226,JUNEAU%2C%20AK,58.3886,-134.133,Alaska
10881,226,Petersburg%2C%20AK,56.774,-132.862,Alaska
11950,226,Dillingham%2C%20AK,59.0622,-158.528,Alaska
12444,226,Cordova%2C%20AK,60.5478,-145.748,Alaska
13884,226,Nome%2C%20AK,64.5092,-165.415,Alaska
14586,226,Unalaska%2C%20AK,53.9345,-166.51,Alaska
14858,226,Soldotna%2C%20AK,60.4875,-151.064,Alaska
14910,226,Kodiak%2C%20AK,57.7985,-152.402,Alaska
16231,226,Kenai%2C%20AK,60.5537,-151.207,Alaska
16770,226,Ouzinkie%2C%20AK,57.9352,-152.458,Alaska
23506,226,Dutch%20Harbor%2C%20AK,53.88,-166.53,Alaska
```

Sequence File

The default output of sqoop is a CSV file (Notice outputs of previous two outputs)
But you can get a different output format such as binary format :

```
sqoop import \
--connect “jdbc:mysql://localhost/training” \
--username training –P \
--table countries \
--target-dir /user/country_binary \
--as-sequencefile
```

Output

Goto : /user/countries_binary go

Go back to dir listing
Advanced view/download options

```
SEQ□!org.apache.hadoop.io.LongWritable  countries□□□□□□□□□□□□t66J□□□
□□□□A□EQUATORIAL GUINEA□GQ□□□□□□□□□□□□□□□□□□□□B□ERITREA□ER□□□□
□□□□F□
FAROE ISLANDS□FO□□□□□□□□□□□□□□□□□□□
```

Free-Form Query

Problem: We need to import data from two tables after merging them as per certain constraints and need certain columns from it

Solution: Free form queries

1) Create two files t1 and t2 as follows

t1

1	USA
2	INDIA
3	CHINA
4	UK

t2

1	3	SHANGHAI
2	1	CHICAGO
3	2	DELHI
4	3	BEIJING
5	2	MUMBAI
6	4	LIVERPOOL
7	1	NEWYORK
8	4	MANCHESHTER
9	1	WASHINGTON
10	4	LONDON

2) Create two tables in mysql and load these tables into them

mysql> create table t1(id int, country_name varchar(20));

mysql> create table t2(number int, id int, city_name varchar(20));

3) Import these data files into created mysql tables.

```
mysql> use training;
Database changed
mysql> create table t1(id int, country_name varchar(20));
Query OK, 0 rows affected (0.01 sec)

mysql> create table t2(number int, id int, city_name varchar(20));
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+--------------------+
| Tables_in_training |
+--------------------+
| Movies             |
| bible_freq         |
| cityByCountry      |
| countries          |
| exported           |
| shake_freq         |
| t1                 |
| t2                 |
+--------------------+
8 rows in set (0.00 sec)

mysql> load data local infile '/home/training/Desktop/t1' into table t1 lines terminated by '\n';
Query OK, 4 rows affected (0.00 sec)
Records: 4  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/training/Desktop/t2' into table t2 lines terminated by '\n';
Query OK, 10 rows affected (0.00 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0
```

Free-Form Query

4) Viewing data from created tables.

*mysql> select * from t1;*

```
mysql> select * from t1;
+------+--------------+
| id   | country_name |
+------+--------------+
|    1 | USA          |
|    2 | INDIA        |
|    3 | CHINA        |
|    4 | UK           |
+------+--------------+
4 rows in set (0.00 sec)
```

5) test sqoop query:

*sqoop import *

*--connect jdbc:mysql://localhost/training *

*--username training -P *

*--query "select * from t2 where \$CONDITIONS" *

--split-by t2.id

--target-dir /user/simple

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2014-04-04 03:54	rw-r--r--	training	supergroup
_logs	dir				2014-04-04 03:54	rwxrwxrwx	training	supergroup
part-m-00000	file	0.04 KB	1	64 MB	2014-04-04 03:54	rw-r--r--	training	supergroup
part-m-00001	file	0.02 KB	1	64 MB	2014-04-04 03:54	rw-r--r--	training	supergroup
part-m-00002	file	0.07 KB	1	64 MB	2014-04-04 03:54	rw-r--r--	training	supergroup

Go back to dir listing

Advanced view/download option

```
3,2,DELHI
5,2,MUMBAI
```

Free-Form Query

Final Code:

*sqoop import *

*--connect jdbc:mysql://localhost/training *

*--username training -P *

*--query "select t2.number, t1.country_name, t2.city_name from t2 join t1 using (id) *

*Where \$CONDITIONS" *

*--split-by id *

*--target-dir /user/join *

-m 1

```
[training@localhost Desktop]$ sqoop import \
> --connect jdbc:mysql://localhost/training \
> --username training -P \
> --query "select t2.number, t1.country_name, t2.city_name from t2 join t1 using (id) where \$CONDITIONS" \
> --split-by id \
> --target-dir /user/join \
> -m 1
Enter password:
14/04/04 04:23:33 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/04/04 04:23:33 INFO tool.CodeGenTool: Beginning code generation
14/04/04 04:23:34 INFO manager.SqlManager: Executing SQL statement: select t2.number, t1.country_name, t2.city_name from t2 join t1 using (id) where  (1 = 0)
14/04/04 04:23:34 INFO manager.SqlManager: Executing SQL statement: select t2.number, t1.country_name, t2.city_name from t2 join t1 using (id) where  (1 = 0)
14/04/04 04:23:34 INFO manager.SqlManager: Executing SQL statement: select t2.number, t1.country_name, t2.city_name from t2 join t1 using (id) where  (1 = 0)
14/04/04 04:23:34 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/b651ca79c6853e85202ba87e9e9a1a1c/QueryResult.java uses or overrides a deprecated API.
```

Free-Form Query

Output:

Contents of directory /user/join

Goto : /user/join go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2014-04-04 04:23	rw-r--r--	training	supergroup
_logs	dir				2014-04-04 04:23	rwxrwxrwx	training	supergroup
part-m-00000	file	0.15 KB	1	64 MB	2014-04-04 04:23	rw-r--r--	training	supergroup

Go back to DFS home

File: /user/join/part-m-00000

Goto : /user/join go

Go back to dir listing
Advanced view/download options

```
1,CHINA,SHANGHAI
2,USA,CHICAGO
3,INDIA,DELHI
4,CHINA,BEIJING
5,INDIA,MUMBAI
6,UK,LIVERPOOL
7,USA,NEWYORK
8,UK,MANCHESHTER
9,USA,WASHINGTON
10,UK,LONDON
```

Exports

- In previous cases, flow of data was from RDBMs to HDFS. Using “export” tool, we can import data from HDFS to RDBMs.
- Before performing export, sqoop fetches table metadata from mysql database. Thus we first need to create a table with required metadata.

1) Table creation in mysql

mysql>Create table table_name(column_name column_type)

```
mysql> create table exported(id int, country varchar(20), code varchar(20));
Query OK, 0 rows affected (0.02 sec)

mysql> describe exported;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| id      | int(11)     | YES  |     | NULL    |       |
| country | varchar(20) | YES  |     | NULL    |       |
| code    | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> 
```

Exports

2) Sqoop query

*sqoop export *

*--connect jdbc:mysql://localhost/training *

*--username training –P *

*--table exported *

*--export-dir /*user/country_imported/part-m-00000

```
[training@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/training --username training -P --table exported --export-dir /user/country_imported/part-m-00000
Enter password:
14/04/04 02:30:41 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/04/04 02:30:41 INFO tool.CodeGenTool: Beginning code generation
14/04/04 02:30:41 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `exported` AS t LIMIT 1
14/04/04 02:30:41 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `exported` AS t LIMIT 1
14/04/04 02:30:41 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/8ea7edb36a09088fcfabf076bbb30af9/exported.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/04/04 02:30:42 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/8ea7edb36a09088fcfabf076bbb30af9/exported.jar
14/04/04 02:30:42 INFO mapreduce.ExportJobBase: Beginning export of exported
14/04/04 02:30:43 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
14/04/04 02:30:43 INFO input.FileInputFormat: Total input paths to process : 1
14/04/04 02:30:43 INFO input.FileInputFormat: Total input paths to process : 1
```

Exports

3) Viewing output in mysql

*mysql> select * from exported*

```
mysql> show tables;
+--------------------+
| Tables_in_training |
+--------------------+
| Movies             |
| bible_freq         |
| cityByCountry      |
| countries          |
| exported           |
| shake_freq         |
+--------------------+
6 rows in set (0.00 sec)

mysql> select * from exported limit 10;
+------+---------------------+------+
| id   | country             | code |
+------+---------------------+------+
|    1 | AFGHANISTAN         | AF   |
|    2 | ALBANIA             | AL   |
|    3 | ALGERIA             | DZ   |
|    4 | AMERICAN SAMOA      | AS   |
|    5 | ANDORRA             | AD   |
|    6 | ANGOLA              | AO   |
|    7 | ANGUILLA            | AI   |
|    8 | ANTARCTICA          | AQ   |
|    9 | ANTIGUA AND BARBUDA | AG   |
|   10 | ARGENTINA           | AR   |
+------+---------------------+------+
10 rows in set (0.00 sec)
```

Integration with hadoop ecosystem

- Till now data was moved between RDBMS to HDFS. This imported data may further be required to be analysed using hive or hbase.
- Sqoop offers property to directly import data to Hive / Hbase.
- Just add “--import-hive” at the end of the command.

```
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/training --username training -P --table countries --target-dir /imported_DB/ --hive-import -m 1
Enter password:
14/04/04 02:52:08 INFO tool.BaseSqoopTool: Using Hive-specific delimiters for output. You can override
14/04/04 02:52:08 INFO tool.BaseSqoopTool: delimiters with --fields-terminated-by, etc.
14/04/04 02:52:08 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/04/04 02:52:08 INFO tool.CodeGenTool: Beginning code generation
14/04/04 02:52:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `countries` AS t LIMIT 1
14/04/04 02:52:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `countries` AS t LIMIT 1
14/04/04 02:52:08 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/7da4d90684b67ed0bcae73ead20f2349/countries.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/04/04 02:52:09 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/7da4d90684b67ed0bcae73ead20f2349/countries.jar
```

Integration with hadoop ecosystem

Viewing in Hive

```
[training@localhost ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
Hive history file=/tmp/training/hive_job_log_training_201404040254_750333817.txt
hive> show databases;
OK
default
try
try2
Time taken: 2.316 seconds
hive> show tables;
OK
countries
Time taken: 0.115 seconds
hive> select * from countries limit 10;
OK
1       AFGHANISTAN     AF
2       ALBANIA AL
3       ALGERIA DZ
4       AMERICAN SAMOA  AS
5       ANDORRA AD
6       ANGOLA  AO
7       ANGUILLA        AI
8       ANTARCTICA      AQ
9       ANTIGUA AND BARBUDA     AG
10      ARGENTINA       AR
Time taken: 0.888 seconds
hive>
```

table is added in default database .

Similarly sqoop also provides commands to import data into Hbase directly.

Thank You!

help@edupristine.com

www.edupristine.com