***Use Case 1: Using Hadoop as a Data Lake with Batch Updates***

In this use case, you will continue to use the MySQL database, but replicate the data into a Hadoop-based data lake. Simply put, a data lake[[1]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "nIfLA) collects data from a bunch of different data sources and puts them all in the same place. A typical data lake will contain data from multiple different sources. This may include data dumps from relational databases with different vendors. It may include semi-structured data such as web server logs.

For this use case, I suggest updating the data in the data lake using a periodic batch process. The batch process can run daily, weekly, or even monthly depending on how often you need to update the data. You can also choose to do incremental updates or full updates.

There are various options available to import MySQL data into a Hadoop cluster. Here are a few options I have used to transfer data from relational databases (Oracle and Postgres) to Hadoop.

* [Sqoop](https://sqoop.apache.org/)is designed as a way to transfer data between Hadoop and structured datastores (e.g. relational databases). In my experience, Sqoop is the best tool to use if you want to reliably replicate an entire database to Hadoop. The Sqoop user guide[[2]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "rjFvu) has information on options for importing data into Hadoop. More than likely, you will want to import the data into Hive or directly into HDFS using a file format like Parquet or Avro.
* If you are comfortable with using Python, Java, or Scala, you can use Spark to read data from JDBC databases, including MySQL[[3]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "IjAzg). I would use this option if you would rather keep the data in a MySQL database and just want to do your processing on Hadoop. You can use it to replicate data, but Sqoop is probably a better choice for replication.
* [Embulk](http://www.embulk.org/docs/) is another open-source bulk data transfer tool. Embulk is a good option if you work with a lot of non-Hadoop, non-database sources. As an example, you could use Embulk to store the history of a Slack channel in a database or on a Hadoop cluster. It has plugins for Amazon S3, MySQL, Postgres, Oracle, Google Cloud Storage, Amazon RedShift, MongoDB, Jira, Google Analytics, Elasticsearch, Slack, Google Spreadsheets, and [many, many more](http://www.embulk.org/plugins/). I have run into a few issues when trying to import binary data types from Oracle, but I do not know if the MySQL plugin has similar issues.

Lastly, you will probably need a scheduler to run the batch jobs. [Oozie](http://oozie.apache.org/" \t "_blank), [Falcon](https://falcon.apache.org/), [Azkaban](https://azkaban.github.io/) are the three most popular Hadoop workflow engines. If the workflow isn't too complicated, you could also use [Cron](https://en.wikipedia.org/wiki/Cron" \t "_blank).

***Use Case 2: Using Hadoop as a Data Lake with Real-Time Updates***

If you want to update the data in near real-time, you will probably want to use something like Kafka[[4]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "Nqpyp). A word of warning, updating data in real-time is more complicated than batch updates. I would suggest avoiding real-time updates unless you absolutely need them as they add a lot of extra complexity.

***Use Case 3: Using Hadoop as a Scalable, Real-time Read/Write Data Store***

Replacing MySQL (or any OLTP database for that matter) with Hadoop is much more complicated than just using Hadoop as a data lake. Migrating from one database to another (e.g. MySQL to Oracle or MySQL to Postgres) is rarely a trivial task. Migrating a traditional OLTP database to Hadoop is even more difficult. I would not recommend migrating to Hadoop unless you are running into significant issues with your existing solution and cannot find a way to solve them within MySQL.

[Apache HBase](https://hbase.apache.org/) and [Apache Accumulo](https://accumulo.apache.org/) are the two open source, NoSQL data stores that are bundled with most Hadoop distributions. Accumulo and HBase are key/value stores and work differently than your typical relational database. [Apache Phoenix](https://phoenix.apache.org/) adds a user-friendly SQL layer on top of HBase but does not abstract away all of the differences between a key/value store and an RDBMS. In other words, do not expect HBase or Accumulo to function as a drop-in replacement for MySQL.

Luckily, if you are replacing MySQL with HBase or Accumulo, you shouldn't need to worry about setting up something to sync the data on a periodic basis. You just need to figure out a way to do a one-time migration of the data. As was the case with batch updates, there a few different ways to migrate the data into HBase or Accumulo.

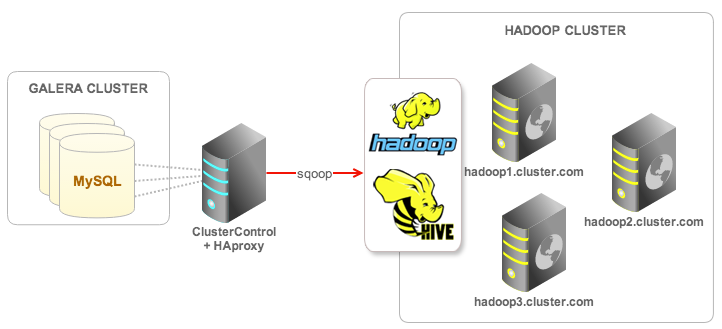
* Sqoop supports importing data into both HBase[[5]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "kDVXF) and Accumulo[[6]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "KEerg). Since they use a different model from a traditional relational database, you may need to make changes to your existing data model.
* Pig also supports storing data into HBase[[7]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "WbNSs) and Accumulo[[8]](https://www.quora.com/How-do-I-migrate-MySQL-to-Hadoop" \l "VbZmf), but it does not support reading from a MySQL database, so you would have to use Sqoop to transfer the data to Hadoop and then use Pig to store it in HBase or Accumulo. You may want to consider this option if you need to transform your data before storing it in HBase or Accumulo.

**Archival and Analytics - Importing MySQL data into Hadoop Cluster using Sqoop**

We won’t bore you with buzzwords like volume, velocity and variety. This post is for MySQL users who want to get their hands dirty with Hadoop, so roll up your sleeves and prepare for work. Why would you ever want to move MySQL data into Hadoop? One good reason is archival and analytics. You might not want to delete old data, but rather move it into Hadoop and make it available for further analysis at a later stage.

In this post, we are going to deploy a Hadoop Cluster and export data in bulk from a Galera Cluster using [Apache Sqoop](http://sqoop.apache.org/docs/1.4.4/SqoopUserGuide.html). Sqoop is a well-proven approach for bulk data loading from a relational database into Hadoop File System. There is also [Hadoop Applier](http://innovating-technology.blogspot.com/) available from [MySQL labs](http://labs.mysql.com/), which works by retrieving INSERT queries from MySQL master binlog and writing them into a file in HDFS in real-time (yes, it applies INSERTs only).

We will use [Apache Ambari](http://ambari.apache.org/) to deploy Hadoop (HDP 2.1) on three servers. We have a clustered Wordpress site running on Galera, and for the purpose of this blog, we will export some user data to Hadoop for archiving purposes. The database name is wordpress, we will use Sqoop to import the data to a Hive table running on HDFS. The following diagram illustrates our setup:



## Downloading Sqoop

We can download the latest version of Sqoop from the following [link](https://sqoop.apache.org/) For this tutorial, we are using version 1.4.5, that is, **sqoop-1.4.5.bin\_\_hadoop-2.0.4-alpha.tar.gz**.

## Step 4: Installing Sqoop

The following commands are used to extract the Sqoop tar ball and move it to “/usr/lib/sqoop” directory.

$tar -xvf sqoop-1.4.4.bin\_\_hadoop-2.0.4-alpha.tar.gz

$ su

password:

# mv sqoop-1.4.4.bin\_\_hadoop-2.0.4-alpha /usr/lib/sqoop

#exit

## Step 5: Configuring bashrc

You have to set up the Sqoop environment by appending the following lines to ~/**.bashrc** file –

Sudo nano ~/.bashrc

#Sqoop

export SQOOP\_HOME=/usr/lib/sqoop

export PATH=$PATH:$SQOOP\_HOME/bin

The following command is used to execute ~/**.bashrc** file.

$ source ~/.bashrc

## Step 6: Configuring Sqoop

To configure Sqoop with Hadoop, you need to edit the **sqoop-env.sh** file, which is placed in the **$SQOOP\_HOME/conf** directory. First of all, Redirect to Sqoop config directory and copy the template file using the following command −

$ cd $SQOOP\_HOME/conf

$ mv sqoop-env-template.sh sqoop-env.sh

Open **sqoop-env.sh** and edit the following lines −

export HADOOP\_COMMON\_HOME=/usr/local/hadoop

export HADOOP\_MAPRED\_HOME=/usr/local/hadoop

## Step 7: Download and Configure mysql-connector-java

We can download **mysql-connector-java-5.1.30.tar.gz** file from the following [link](http://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/).

The following commands are used to extract mysql-connector-java tarball and move **mysql-connector-java-5.1.30-bin.jar** to /usr/lib/sqoop/lib directory.

$ tar -zxf mysql-connector-java-5.1.30.tar.gz

$ su

password:

# cd mysql-connector-java-5.1.30

# mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib

## Step 8: Verifying Sqoop

The following command is used to verify the Sqoop version.

$ cd $SQOOP\_HOME/bin

$ sqoop -version

Expected output −

14/12/17 14:52:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5

Sqoop 1.4.5 git commit id 5b34accaca7de251fc91161733f906af2eddbe83

Compiled by abe on Fri Aug 1 11:19:26 PDT 2014

Sqoop installation is complete.

his chapter describes how to import data from MySQL database to Hadoop HDFS. The ‘Import tool’ imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in the text files or as binary data in Avro and Sequence files.

## Syntax

The following syntax is used to import data into HDFS.

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

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

### Example

Let us take an example of three tables named as **emp**, **emp\_add**, and **emp\_contact**, which are in a database called userdb in a MySQL database server.

The three tables and their data are as follows.

### emp:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **Name** | **deg** | **salary** | **dept** |
| 1201 | Gopal | manager | 50,000 | TP |
| 1202 | Manisha | Proof reader | 50,000 | TP |
| 1203 | khalil | php dev | 30,000 | AC |
| 1204 | prasanth | php dev | 30,000 | AC |
| 1204 | kranthi | admin | 20,000 | TP |

### emp\_add:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **hno** | **street** | **city** |
| 1201 | 288A | vgiri | jublee |
| 1202 | 108I | aoc | sec-bad |
| 1203 | 144Z | pgutta | hyd |
| 1204 | 78B | old city | sec-bad |
| 1205 | 720X | hitec | sec-bad |

### emp\_contact:

|  |  |  |
| --- | --- | --- |
| **id** | **phno** | **email** |
| 1201 | 2356742 | gopal@tp.com |
| 1202 | 1661663 | manisha@tp.com |
| 1203 | 8887776 | khalil@ac.com |
| 1204 | 9988774 | prasanth@ac.com |
| 1205 | 1231231 | kranthi@tp.com |

## Importing a Table

Sqoop tool ‘import’ is used to import table data from the table to the Hadoop file system as a text file or a binary file.

The following command is used to import the **emp** table from MySQL database server to HDFS.

$ sqoop import \

--connect jdbc:mysql://localhost/userdb \

--username root \

--table emp --m 1

If it is executed successfully, then you get the following output.

14/12/22 15:24:54 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5

14/12/22 15:24:56 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

14/12/22 15:24:56 INFO tool.CodeGenTool: Beginning code generation

14/12/22 15:24:58 INFO manager.SqlManager: Executing SQL statement:

SELECT t.\* FROM `emp` AS t LIMIT 1

14/12/22 15:24:58 INFO manager.SqlManager: Executing SQL statement:

SELECT t.\* FROM `emp` AS t LIMIT 1

14/12/22 15:24:58 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /usr/local/hadoop

14/12/22 15:25:11 INFO orm.CompilationManager: Writing jar file:

/tmp/sqoop-hadoop/compile/cebe706d23ebb1fd99c1f063ad51ebd7/emp.jar

-----------------------------------------------------

-----------------------------------------------------

14/12/22 15:25:40 INFO mapreduce.Job: The url to track the job:

http://localhost:8088/proxy/application\_1419242001831\_0001/

14/12/22 15:26:45 INFO mapreduce.Job: Job job\_1419242001831\_0001 running in uber mode :

false

14/12/22 15:26:45 INFO mapreduce.Job: map 0% reduce 0%

14/12/22 15:28:08 INFO mapreduce.Job: map 100% reduce 0%

14/12/22 15:28:16 INFO mapreduce.Job: Job job\_1419242001831\_0001 completed successfully

-----------------------------------------------------

-----------------------------------------------------

14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Transferred 145 bytes in 177.5849 seconds

(0.8165 bytes/sec)

14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Retrieved 5 records.

To verify the imported data in HDFS, use the following command.

$ $HADOOP\_HOME/bin/hadoop fs -cat /emp/part-m-\*

It shows you the **emp** table data and fields are separated with comma (,).

1201, gopal, manager, 50000, TP

1202, manisha, preader, 50000, TP

1203, kalil, php dev, 30000, AC

1204, prasanth, php dev, 30000, AC

1205, kranthi, admin, 20000, TP

## Importing into Target Directory

We can specify the target directory while importing table data into HDFS using the Sqoop import tool.

Following is the syntax to specify the target directory as option to the Sqoop import command.

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

The following command is used to import **emp\_add** table data into ‘/queryresult’ directory.

$ sqoop import \

--connect jdbc:mysql://localhost/userdb \

--username root \

--table emp\_add \

--m 1 \

--target-dir /queryresult

The following command is used to verify the imported data in /queryresult directory form **emp\_add** table.

$ $HADOOP\_HOME/bin/hadoop fs -cat /queryresult/part-m-\*

It will show you the emp\_add table data with comma (,) separated fields.

1201, 288A, vgiri, jublee

1202, 108I, aoc, sec-bad

1203, 144Z, pgutta, hyd

1204, 78B, oldcity, sec-bad

1205, 720C, hitech, sec-bad

## Import Subset of Table Data

We can import a subset of a table using the ‘where’ clause in Sqoop import tool. It executes the corresponding SQL query in the respective database server and stores the result in a target directory in HDFS.

The syntax for where clause is as follows.

--where <condition>

The following command is used to import a subset of **emp\_add** table data. The subset query is to retrieve the employee id and address, who lives in Secunderabad city.

$ sqoop import \

--connect jdbc:mysql://localhost/userdb \

--username root \

--table emp\_add \

--m 1 \

--where “city =’sec-bad’” \

--target-dir /wherequery

The following command is used to verify the imported data in /wherequery directory from the **emp\_add** table.

$ $HADOOP\_HOME/bin/hadoop fs -cat /wherequery/part-m-\*

It will show you the **emp\_add** table data with comma (,) separated fields.

1202, 108I, aoc, sec-bad

1204, 78B, oldcity, sec-bad

1205, 720C, hitech, sec-bad

## Incremental Import

Incremental import is a technique that imports only the newly added rows in a table. It is required to add ‘incremental’, ‘check-column’, and ‘last-value’ options to perform the incremental import.

The following syntax is used for the incremental option in Sqoop import command.

--incremental <mode>

--check-column <column name>

--last value <last check column value>

Let us assume the newly added data into **emp** table is as follows −

1206, satish p, grp des, 20000, GR

The following command is used to perform the incremental import in the **emp**table.

$ sqoop import \

--connect jdbc:mysql://localhost/userdb \

--username root \

--table emp \

--m 1 \

--incremental append \

--check-column id \

-last value 1205

The following command is used to verify the imported data from **emp** table to HDFS emp/ directory.

$ $HADOOP\_HOME/bin/hadoop fs -cat /emp/part-m-\*

It shows you the **emp** table data with comma (,) separated fields.

1201, gopal, manager, 50000, TP

1202, manisha, preader, 50000, TP

1203, kalil, php dev, 30000, AC

1204, prasanth, php dev, 30000, AC

1205, kranthi, admin, 20000, TP

1206, satish p, grp des, 20000, GR

The following command is used to see the modified or newly added rows from the **emp** table.

$ $HADOOP\_HOME/bin/hadoop fs -cat /emp/part-m-\*1

It shows you the newly added rows to the **emp** table with comma (,) separated fields.

1206, satish p, grp des, 20000, GR

# Import All Tables

This chapter describes how to import all the tables from the RDBMS database server to the HDFS. Each table data is stored in a separate directory and the directory name is same as the table name.

## Syntax

The following syntax is used to import all tables.

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

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

### Example

Let us take an example of importing all tables from the **userdb** database. The list of tables that the database **userdb** contains is as follows.

+--------------------+

| Tables |

+--------------------+

| emp |

| emp\_add |

| emp\_contact |

+--------------------+

The following command is used to import all the tables from the **userdb**database.

$ sqoop import-all-tables \

--connect jdbc:mysql://localhost/userdb \

--username root

**Note** − If you are using the import-all-tables, it is mandatory that every table in that database must have a primary key field.

The following command is used to verify all the table data to the userdb database in HDFS.

$ $HADOOP\_HOME/bin/hadoop fs -ls

It will show you the list of table names in userdb database as directories.

### Output

drwxr-xr-x - hadoop supergroup 0 2014-12-22 22:50 \_sqoop

drwxr-xr-x - hadoop supergroup 0 2014-12-23 01:46 emp

drwxr-xr-x - hadoop supergroup 0 2014-12-23 01:50 emp\_add

drwxr-xr-x - hadoop supergroup 0 2014-12-23 01:52 emp\_contact

# Sqoop - Export

This chapter describes how to export data back from the HDFS to the RDBMS database. The target table must exist in the target database. The files which are given as input to the Sqoop contain records, which are called rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.

The default operation is to insert all the record from the input files to the database table using the INSERT statement. In update mode, Sqoop generates the UPDATE statement that replaces the existing record into the database.

## Syntax

The following is the syntax for the export command.

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

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

### Example

Let us take an example of the employee data in file, in HDFS. The employee data is available in **emp\_data** file in ‘emp/’ directory in HDFS. The **emp\_data**is as follows.

1201, gopal, manager, 50000, TP

1202, manisha, preader, 50000, TP

1203, kalil, php dev, 30000, AC

1204, prasanth, php dev, 30000, AC

1205, kranthi, admin, 20000, TP

1206, satish p, grp des, 20000, GR

It is mandatory that the table to be exported is created manually and is present in the database from where it has to be exported.

The following query is used to create the table ‘employee’ in mysql command line.

$ mysql

mysql> USE db;

mysql> CREATE TABLE employee (

id INT NOT NULL PRIMARY KEY,

name VARCHAR(20),

deg VARCHAR(20),

salary INT,

dept VARCHAR(10));

The following command is used to export the table data (which is in **emp\_data** file on HDFS) to the employee table in db database of Mysql database server.

$ sqoop export \

--connect jdbc:mysql://localhost/db \

--username root \

--table employee \

--export-dir /emp/emp\_data

The following command is used to verify the table in mysql command line.

mysql>select \* from employee;

If the given data is stored successfully, then you can find the following table of given employee data.

+------+--------------+-------------+-------------------+--------+

| Id | Name | Designation | Salary | Dept |

+------+--------------+-------------+-------------------+--------+

| 1201 | gopal | manager | 50000 | TP |

| 1202 | manisha | preader | 50000 | TP |

| 1203 | kalil | php dev | 30000 | AC |

| 1204 | prasanth | php dev | 30000 | AC |

| 1205 | kranthi | admin | 20000 | TP |

| 1206 | satish p | grp des | 20000 | GR |

+------+--------------+-------------+-------------------+--------+

# Sqoop - Job

This chapter describes how to create and maintain the Sqoop jobs. Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job. This re-calling or re-executing is used in the incremental import, which can import the updated rows from RDBMS table to HDFS.

## Syntax

The following is the syntax for creating a Sqoop job.

$ sqoop job (generic-args) (job-args)

[-- [subtool-name] (subtool-args)]

$ sqoop-job (generic-args) (job-args)

[-- [subtool-name] (subtool-args)]

## Create Job (--create)

Here we are creating a job with the name **myjob**, which can import the table data from RDBMS table to HDFS. The following command is used to create a job that is importing data from the **employee** table in the **db** database to the HDFS file.

$ sqoop job --create myjob \

-- import \

--connect jdbc:mysql://localhost/db \

--username root \

--table employee --m 1

## Verify Job (--list)

**‘--list’** argument is used to verify the saved jobs. The following command is used to verify the list of saved Sqoop jobs.

$ sqoop job --list

It shows the list of saved jobs.

Available jobs:

myjob

## Inspect Job (--show)

**‘--show’** argument is used to inspect or verify particular jobs and their details. The following command and sample output is used to verify a job called **myjob**.

$ sqoop job --show myjob

It shows the tools and their options, which are used in **myjob**.

Job: myjob

Tool: import Options:

----------------------------

direct.import = true

codegen.input.delimiters.record = 0

hdfs.append.dir = false

db.table = employee

...

incremental.last.value = 1206

...

## Execute Job (--exec)

**‘--exec’** option is used to execute a saved job. The following command is used to execute a saved job called **myjob**.

$ sqoop job --exec myjob

It shows you the following output.

10/08/19 13:08:45 INFO tool.CodeGenTool: Beginning code generation

...

# Sqoop - Codegen

This chapter describes the importance of ‘codegen’ tool. From the viewpoint of object-oriented application, every database table has one DAO class that contains ‘getter’ and ‘setter’ methods to initialize objects. This tool (-codegen) generates the DAO class automatically.

It generates DAO class in Java, based on the Table Schema structure. The Java definition is instantiated as a part of the import process. The main usage of this tool is to check if Java lost the Java code. If so, it will create a new version of Java with the default delimiter between fields.

## Syntax

The following is the syntax for Sqoop codegen command.

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

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

### Example

Let us take an example that generates Java code for the **emp** table in the **userdb** database.

The following command is used to execute the given example.

$ sqoop codegen \

--connect jdbc:mysql://localhost/userdb \

--username root \

--table emp

If the command executes successfully, then it will produce the following output on the terminal.

14/12/23 02:34:40 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5

14/12/23 02:34:41 INFO tool.CodeGenTool: Beginning code generation

……………….

14/12/23 02:34:42 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /usr/local/hadoop

Note: /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/emp.java uses or

overrides a deprecated API.

Note: Recompile with -Xlint:deprecation for details.

14/12/23 02:34:47 INFO orm.CompilationManager: Writing jar file:

/tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/emp.jar

### Verification

Let us take a look at the output. The path, which is in bold, is the location that the Java code of the **emp** table generates and stores. Let us verify the files in that location using the following commands.

$ cd /tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/

$ ls

emp.class

emp.jar

emp.java

If you want to verify in depth, compare the **emp** table in the **userdb** database and **emp.java** in the following directory

/tmp/sqoop-hadoop/compile/9a300a1f94899df4a9b10f9935ed9f91/.

# Sqoop - Eval

This chapter describes how to use the Sqoop ‘eval’ tool. It allows users to execute user-defined queries against respective database servers and preview the result in the console. So, the user can expect the resultant table data to import. Using eval, we can evaluate any type of SQL query that can be either DDL or DML statement.

## Syntax

The following syntax is used for Sqoop eval command.

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

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

## Select Query Evaluation

Using eval tool, we can evaluate any type of SQL query. Let us take an example of selecting limited rows in the **employee** table of **db** database. The following command is used to evaluate the given example using SQL query.

$ sqoop eval \

--connect jdbc:mysql://localhost/db \

--username root \

--query “SELECT \* FROM employee LIMIT 3”

If the command executes successfully, then it will produce the following output on the terminal.

+------+--------------+-------------+-------------------+--------+

| Id | Name | Designation | Salary | Dept |

+------+--------------+-------------+-------------------+--------+

| 1201 | gopal | manager | 50000 | TP |

| 1202 | manisha | preader | 50000 | TP |

| 1203 | khalil | php dev | 30000 | AC |

+------+--------------+-------------+-------------------+--------+

## Insert Query Evaluation

Sqoop eval tool can be applicable for both modeling and defining the SQL statements. That means, we can use eval for insert statements too. The following command is used to insert a new row in the **employee** table of **db**database.

$ sqoop eval \

--connect jdbc:mysql://localhost/db \

--username root \

-e “INSERT INTO employee VALUES(1207,‘Raju’,‘UI dev’,15000,‘TP’)”

If the command executes successfully, then it will display the status of the updated rows on the console.

Or else, you can verify the employee table on MySQL console. The following command is used to verify the rows of **employee** table of **db** database using select’ query.

mysql>

mysql> use db;

mysql> SELECT \* FROM employee;

+------+--------------+-------------+-------------------+--------+

| Id | Name | Designation | Salary | Dept |

+------+--------------+-------------+-------------------+--------+

| 1201 | gopal | manager | 50000 | TP |

| 1202 | manisha | preader | 50000 | TP |

| 1203 | khalil | php dev | 30000 | AC |

| 1204 | prasanth | php dev | 30000 | AC |

| 1205 | kranthi | admin | 20000 | TP |

| 1206 | satish p | grp des | 20000 | GR |

| 1207 | Raju | UI dev | 15000 | TP |

+------+--------------+-------------+-------------------+--------+

# Sqoop - List Databases

This chapter describes how to list out the databases using Sqoop. Sqoop list-databases tool parses and executes the ‘SHOW DATABASES’ query against the database server. Thereafter, it lists out the present databases on the server.

## Syntax

The following syntax is used for Sqoop list-databases command.

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

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

## Sample Query

The following command is used to list all the databases in the MySQL database server.

$ sqoop list-databases \

--connect jdbc:mysql://localhost/ \

--username root

If the command executes successfully, then it will display the list of databases in your MySQL database server as follows.

...

13/05/31 16:45:58 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

mysql

test

userdb

db

# Sqoop - List Tables

This chapter describes how to list out the tables of a particular database in MySQL database server using Sqoop. Sqoop list-tables tool parses and executes the ‘SHOW TABLES’ query against a particular database. Thereafter, it lists out the present tables in a database.

## Syntax

The following syntax is used for Sqoop list-tables command.

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

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

## Sample Query

The following command is used to list all the tables in the **userdb** database of MySQL database server.

$ sqoop list-tables \

--connect jdbc:mysql://localhost/userdb \

--username root

If the command is executes successfully, then it will display the list of tables in the **userdb** database as follows.

...

13/05/31 16:45:58 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

emp

emp\_add

emp\_contact