

Principles Sqoop:

Name: อวชัย ภิรมย์รัตน์

Tel. : 086-813-5354

e-mail : p.Auoychai@gmail.com

Big Data

What is Sqoop :

SQL-to-Hadoop:

As a tools for Transferring bulk data between Hadoop and Relational DB



RDBMS – Hadoop Data Transferring :

➤ Use-Case :

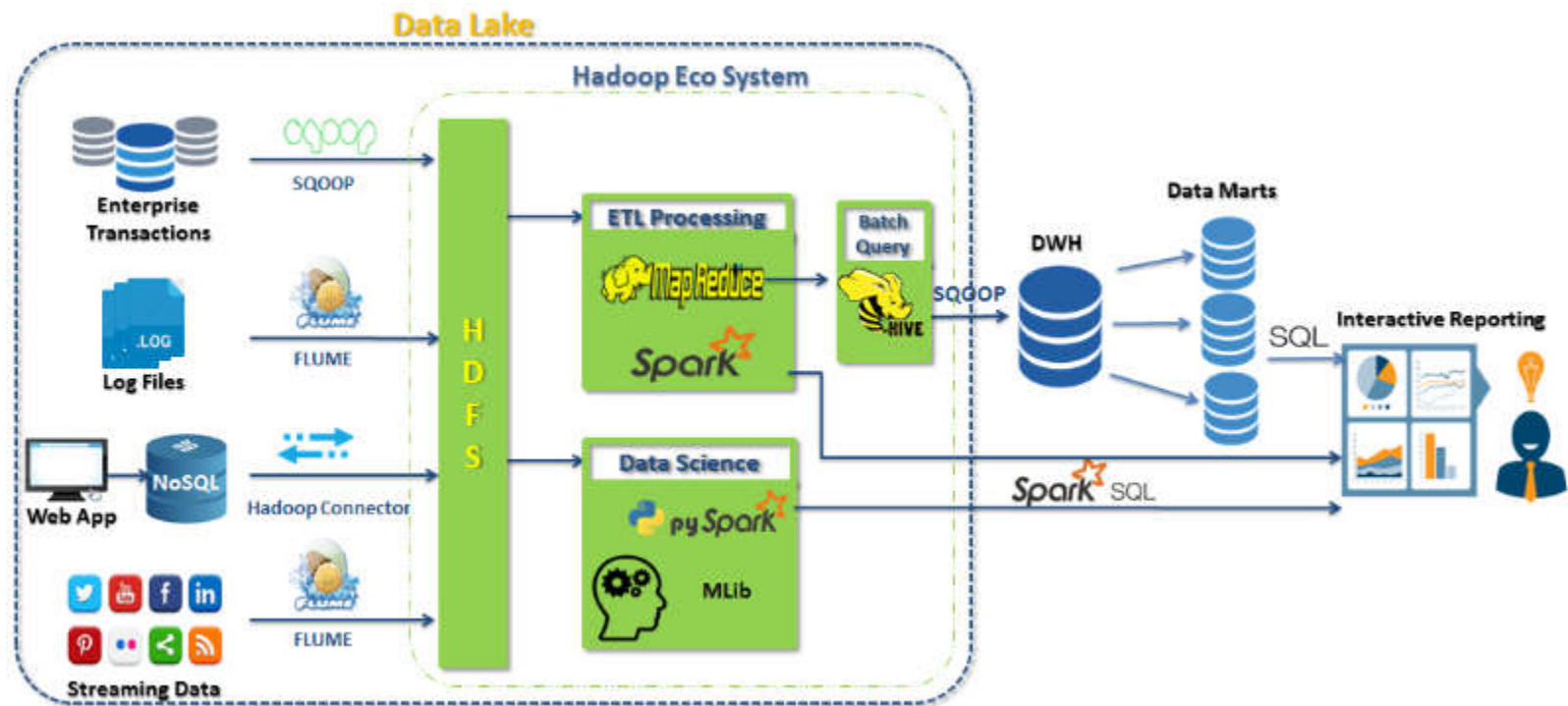
USE CASE #2: ETL FOR DWH

- Transform operational data for data warehouse reports in Hadoop as the batch transformation “engine”



RDBMS – Hadoop Data Transferring :

➤ Use-Case :



Why needed Sqoop :

- As more organizations deploy Hadoop to analyze vast streams of information, they may find they need to transfer large amount of data between Hadoop and their existing databases, data warehouses and other data sources
- Loading bulk data into Hadoop from production systems or accessing it from map-reduce applications running on a large cluster is a challenging task since transferring data using scripts is a inefficient and time-consuming task

Hadoop – Sqoop :

- Hadoop is great for storing massive data in terms of volume using HDFS
- It Provides a scalable processing environment for structured and unstructured data
- But it's Batch-Oriented and thus not suitable for low latency interactive query operations
- Sqoop is basically an ETL Tool used to copy data between HDFS and SQL databases
 - Import SQL data to HDFS for archival or analysis
 - Export HDFS to SQL (e.g : summarized data used in a DW fact table)

What Sqoop Does :

- Designed to efficiently transfer bulk data between Apache Hadoop and structured datastores such as relational databases, Apache Sqoop:
- Allows data imports from external datastores and enterprise data warehouses into Hadoop
- Parallelizes data transfer for fast performance and optimal system utilization
- Copies data quickly from external systems to Hadoop
- Makes data analysis more efficient
- Mitigates excessive loads to external systems.

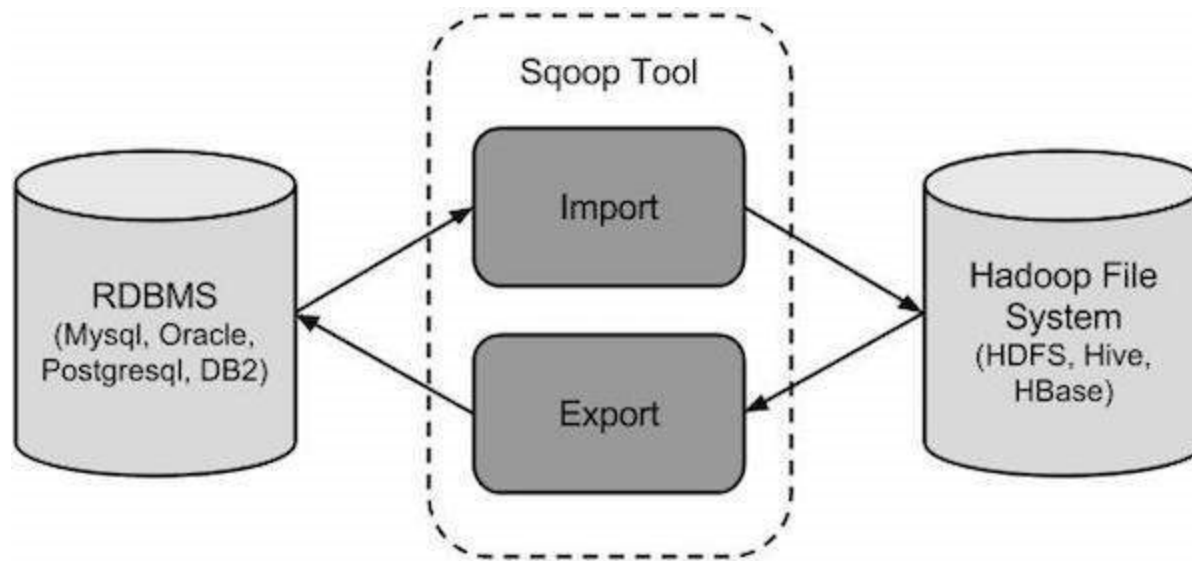
How Sqoop Works :

- Sqoop provides a pluggable connector mechanism for optimal connectivity to external systems.
- The Sqoop extension API provides a convenient framework for building new connectors which can be dropped into Sqoop installations to provide connectivity to various systems.
- Sqoop itself comes bundled with various connectors that can be used for popular database and data warehousing systems.

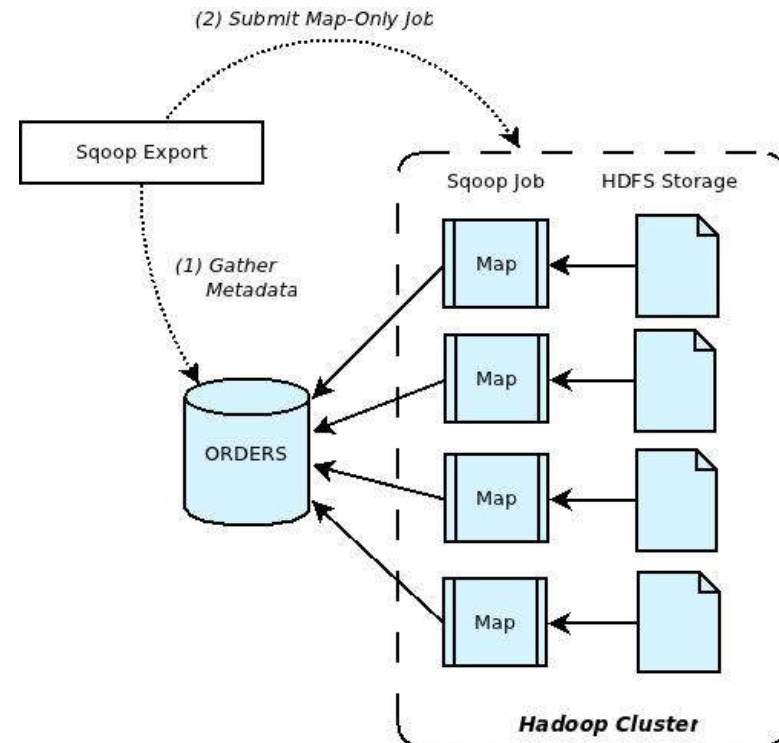
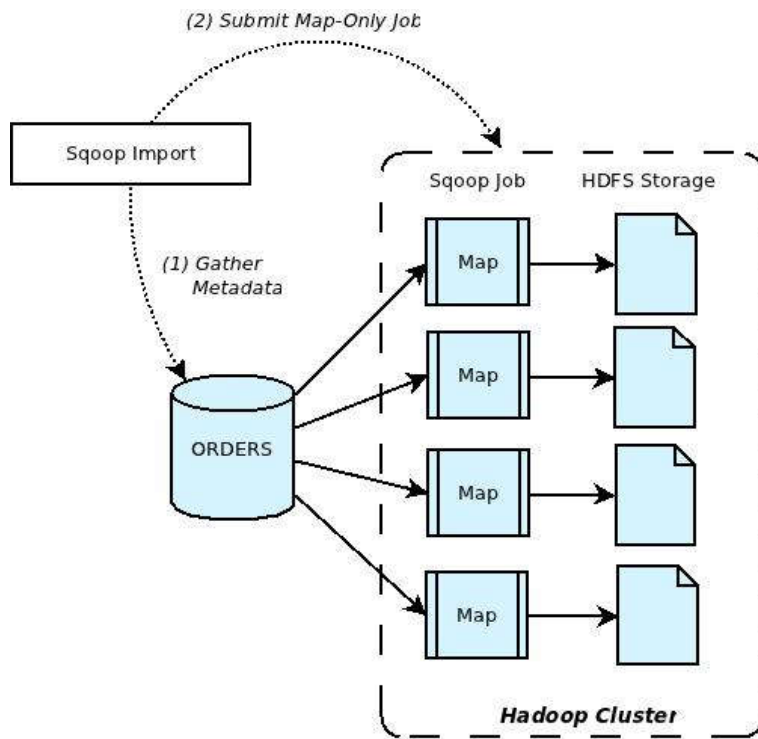
Who Uses Sqoop ? :

- Online Marketer Coupons.com uses sqoop to exchange data between Hadoop and the IBM Netezza data warehouse appliance, The organization can query its structured databases and pipe the results into Hadoop using sqoop.
- Education company The Apollo group also uses the software not only to extract data from databases but to inject the results from Hadoop jobs back into relational databases
- And countless other hadoop users use sqoop to efficiently move their data

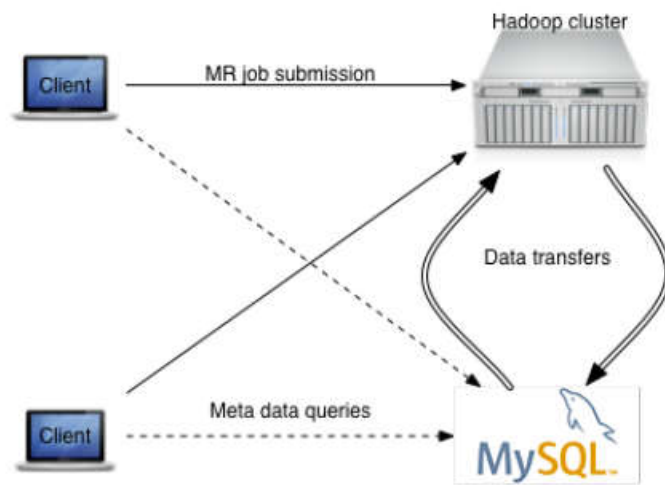
Sqoop Architecture :



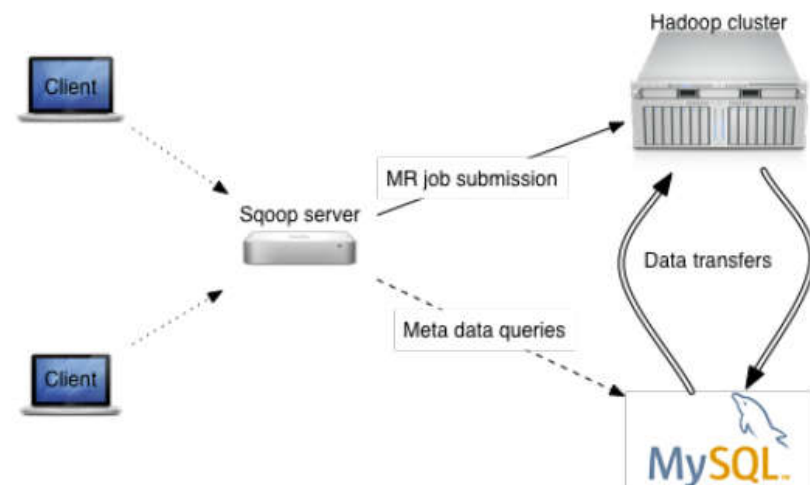
Sqoop Operation Process :



Sqoop Architecture :



Sqoop1



Sqoop2

Sqoop Architecture :

Feature	Sqoop 1	Sqoop 2
Connectors for all major RDBMS	Supported.	Not supported. Workaround: Use the generic JDBC Connector which has been tested on the following databases: Microsoft SQL Server, PostgreSQL, MySQL and Oracle. This connector should work on any other JDBC compliant database. However, performance might not be comparable to that of specialized connectors in Sqoop.
Kerberos Security Integration	Supported.	Supported.
Data transfer from RDBMS to Hive or HBase	Supported.	Not supported. 1.Workaround: Follow this two-step approach. Import data from RDBMS into HDFS 2.Load data into Hive or HBase manually using appropriate tools and commands such as the LOAD DATA statement in Hive
Data transfer from Hive or HBase to RDBMS	1.Not supported. Workaround: Follow this two-step approach. Extract data from Hive or HBase into HDFS (either as a text or Avro file) 2.Use Sqoop to export output of previous step to RDBMS	Not supported. Follow the same workaround as for Sqoop 1.

What Sqoop Does :

- Designed to efficiently transfer bulk data between Apache Hadoop and structured datastores such as relational databases, Apache Sqoop:
- **Allows data imports** from external datastores and enterprise data warehouses into Hadoop
- **Parallelizes data transfer** for fast performance and optimal system utilization
- **Copies data quickly** from external systems to Hadoop
- **Makes data analysis more efficient**
- **Mitigates excessive loads** to external systems.

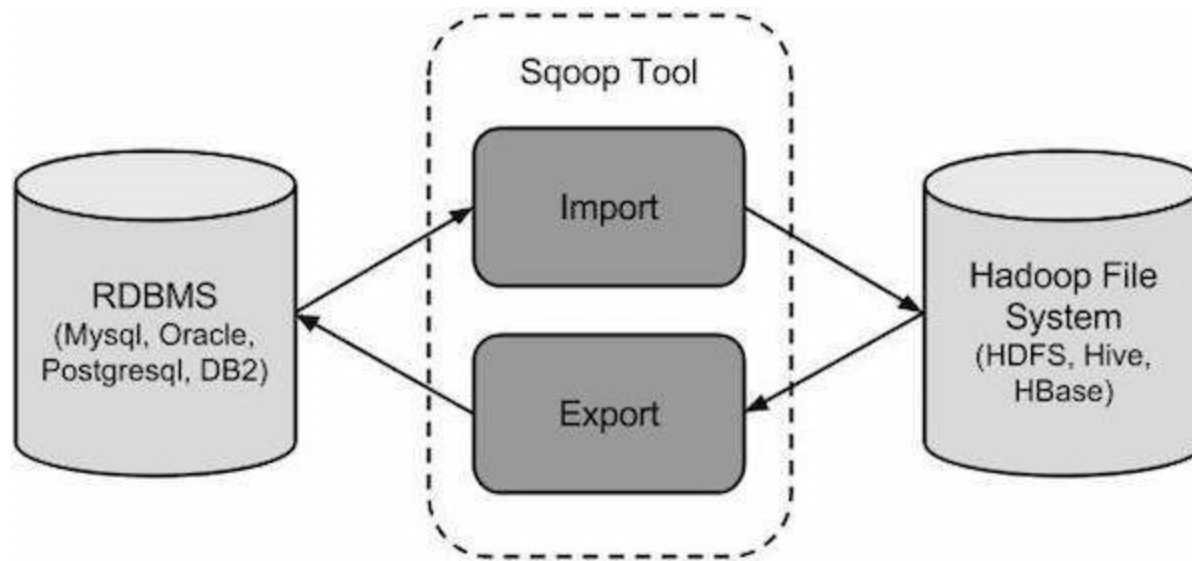
How Sqoop Work :

- Sqoop provides a pluggable connector mechanism for optimal connectivity to external systems.
- The Sqoop extension API provides a convenient framework for building new connectors which can be dropped into Sqoop installations to provide connectivity to various systems.
- Sqoop itself comes bundled with various connectors that can be used for popular database and data warehousing systems.

Who Uses Sqoop :

- Online Marketer *Coupons.com* uses sqoop to exchange data between Hadoop and the IBM Netezza data warehouse appliance, The organization can query its structured databases and pipe the results into Hadoop using sqoop.
- Education company *The Apollo group* also uses the software not only to extract data from databases but to inject the results from Hadoop jobs back into relational databases
- And countless other hadoop users use sqoop to efficiently move their data

Sqoop Data Transferring Operation :



Sqoop Data Transferring Operation :

- Sqoop Command for Manual data operation.
- Sqoop Job for Automatic data management workflow.

User Guide:

https://sqoop.apache.org/docs/1.4.3/SqoopUserGuide.html#_controlling_the_hadoop_installation

Sqoop Operation : View DB Structure

Connect databases

List Database Structure

Operation – Import / Export

Sqoop Operation : Show DB Structure

List databases

```
$ sqoop list-databases --connect jdbc:mysql://<mysql-server>/  
--username <username> --password <password>
```

List tables

```
$ sqoop list-tables --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password>
```

Sqoop Operation : Show DB Structure

List databases

```
$ sqoop list-databases --connect jdbc:mysql://<mysql-server>/  
--username <username> --password <password>
```

```
auoychai@ubtserver:~$ sqoop list-databases --connect jdbc:mysql://192.168.1.34/ --username auoychai  
--password 123456  
17/06/04 14:00:55 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6  
17/06/04 14:00:55 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Co  
nsider using -P instead.  
17/06/04 14:00:55 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
information_schema  
employees  
mysql  
performance_schema  
sys  
auoychai@ubtserver:~$ _
```

Sqoop Operation : Show DB Structure

List tables

\$ sqoop list-tables --connect jdbc:mysql://<mysql-server>/<db-name>

--username <username> --password <password>

```
auoychai@ubtserver:~$ sqoop list-tables --connect jdbc:mysql://192.168.1.34/employees --username auoychai --password 123456
17/06/04 13:59:29 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
17/06/04 13:59:29 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
17/06/04 13:59:29 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
current_dept_emp
departments
dept_emp
dept_emp_latest_date
dept_manager
employees
employees1
employees2
employees3
salaries
titles
auoychai@ubtserver:~$
```

Sqoop Operation : Import Data

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>
```

```
--username <username> --password <password>
```

Table 3. Import control arguments:

Argument	Description
<code>--append</code>	Append data to an existing dataset in HDFS
<code>--as-avrodatafile</code>	Imports data to Avro Data Files
<code>--as-sequencefile</code>	Imports data to SequenceFiles
<code>--as-textfile</code>	Imports data as plain text (default)
<code>--boundary-query <statement></code>	Boundary query to use for creating splits
<code>--columns <col,col,col...></code>	Columns to import from table
<code>--direct</code>	Use direct import fast path
<code>--direct-split-size <n></code>	Split the input stream every <i>n</i> bytes when importing in direct mode
<code>--fetch-size <n></code>	Number of entries to read from database at once.
<code>--inline-lob-limit <n></code>	Set the maximum size for an inline LOB
<code>-m, --num-mappers <n></code>	Use <i>n</i> map tasks to import in parallel
<code>-e, --query <statement></code>	Import the results of <i>statement</i> .
<code>--split-by <column-name></code>	Column of the table used to split work units
<code>--table <table-name></code>	Table to read
<code>--target-dir <dir></code>	HDFS destination dir
<code>--warehouse-dir <dir></code>	HDFS parent for table destination
<code>--where <where clause></code>	WHERE clause to use during import
<code>-z, --compress</code>	Enable compression
<code>--compression-codec <c></code>	Use Hadoop codec (default gzip)
<code>--null-string <null-string></code>	The string to be written for a null value for string columns
<code>--null-non-string <null-string></code>	The string to be written for a null value for non-string columns

Import Data to HDFS :

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password> --table <table-name> --m  
<n> --warehouse-dir <hdfs-path>
```

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password> --table <table-name> --m  
<n> -- target-dir <hdfs-path>
```


Import Data to HDFS :

Query Condition:

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>
```

```
--username <username> --password <password>
```

```
--query “ <sql-select-statement> where $conditions ” --split-by <column>
```

```
-- m <n> -- target-dir <hdfs-path>
```

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>
```

```
--username <username> --password <password>
```

```
--table <table-name> --where <condition> --split-by <column> -- m <n> -
```

```
- target-dir <hdfs-path>
```

Import to Hive :

Query Condition:

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>
```

```
--username <username> --password <password>
```

```
--hive-import --hive-table <dbname.tablename> --m <n>
```

Export HDFS Data to RDB :

Table 18. Common arguments

Argument	Description
<code>--connect <jdbc-uri></code>	Specify JDBC connect string
<code>--connection-manager <class-name></code>	Specify connection manager class to use
<code>--driver <class-name></code>	Manually specify JDBC driver class to use
<code>--hadoop-mapred-home <dir></code>	Override \$HADOOP_MAPRED_HOME
<code>--help</code>	Print usage instructions
<code>-P</code>	Read password from console
<code>--password <password></code>	Set authentication password
<code>--username <username></code>	Set authentication username
<code>--verbose</code>	Print more information while working
<code>--connection-param-file <filename></code>	Optional properties file that provides connection parameters

Table 20. Export control arguments:

Argument	Description
<code>--direct</code>	Use direct export fast path
<code>--export-dir <dir></code>	HDFS source path for the export
<code>-m, --num-mappers <n></code>	Use <i>n</i> map tasks to export in parallel
<code>--table <table-name></code>	Table to populate
<code>--call <stored-proc-name></code>	Stored Procedure to call
<code>--update-key <col-name></code>	Anchor column to use for updates. Use a comma separated list of columns if there are more than one column.
<code>--update-mode <mode></code>	Specify how updates are performed when new rows are found with non-matching keys in database. Legal values for <i>mode</i> include <code>updateonly</code> (default) and <code>allowinsert</code> .
<code>--input-null-string <null-string></code>	The string to be interpreted as null for string columns
<code>--input-null-non-string <null-string></code>	The string to be interpreted as null for non-string columns
<code>--staging-table <staging-table-name></code>	The table in which data will be staged before being inserted into the destination table.
<code>--clear-staging-table</code>	Indicates that any data present in the staging table can be deleted.
<code>--batch</code>	Use batch mode for underlying statement execution.

Export HDFS Data to RDB :

Insert :

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password>  
--table <target-table> -- export-dir <hdfs-path>
```

#Insert/Update :

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password>  
--table <target-table> --update-key <TB-PK> -- export-dir <hdfs-path>
```

Sqoop Job :

Query Condition:

```
$ sqoop job --create <job-name-id> -- < Sqoop command >
```

```
$ sqoop job --list
```

```
$ sqoop job --show <job-name-id>
```

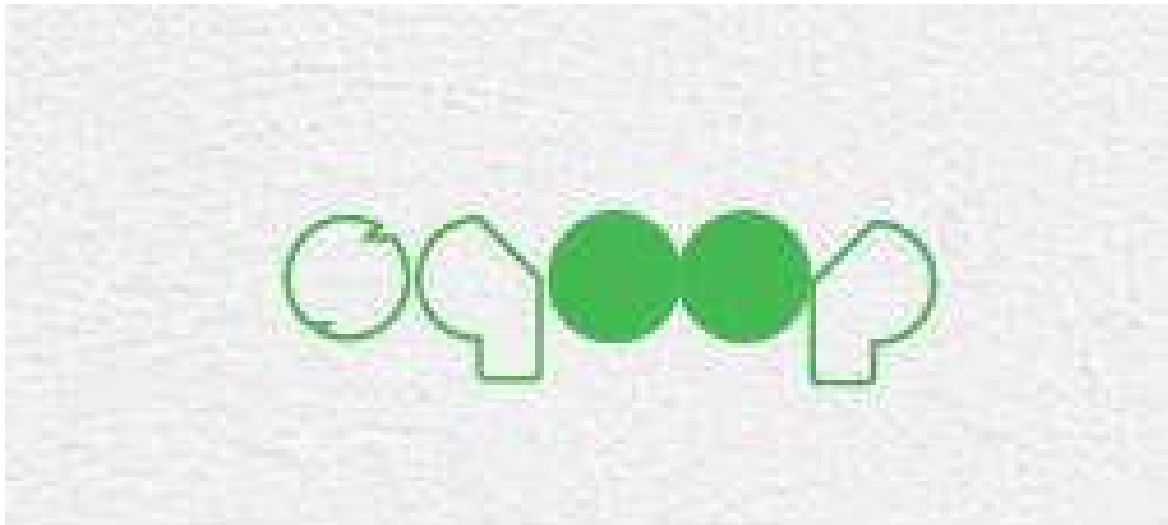
```
$ sqoop job --exec <job-name-id>
```

```
$ sqoop job --delete <job-name-id>
```

Table 24. Job management options:

Argument	Description
<code>--create</code> <code><job-id></code>	Define a new saved job with the specified job-id (name). A second Sqoop command-line, separated by a <code>--</code> should be specified; this defines the saved job.
<code>--delete</code> <code><job-id></code>	Delete a saved job.
<code>--exec <job-id></code>	Given a job defined with <code>--create</code> , run the saved job.
<code>--show <job-id></code>	Show the parameters for a saved job.
<code>--list</code>	List all saved jobs

Hand-Ons:



Installation

Installation :

Download & Install Package :

```
$wget http://www-us.apache.org/sqoop-1.4.6.bin_hadoop-0.23.tar.gz
```

```
$tar xvf sqoop-1.4.6.bin_hadoop-0.23.tar.gz
```

```
$mv sqoop-1.4.6.bin_hadoop-0.23 /usr/local/sqoop
```

```
$chown -R auoychai:auoychai /usr/local/sqoop
```

Set Environment Variable :

```
$nano .bashrc
```

```
export SQOOP_HOME=/usr/local/sqoop
```

```
export PATH=$PATH:$SQOOP_HOME/bin
```

```
$source .bashrc
```


Installation :

Configuration Sqoop :

```
$cp /usr/local/sqoop/conf/sqoop-env-template.sh /usr/local/sqoop/conf/sqoop-  
env.sh
```

Edit sqoop-env.sh :

```
$cd /usr/local/sqoop/conf
```

```
$nano sqoop-env.sh
```

```
export HADOOP_COMMAND_HOME=/usr/local/hadoop
```

```
export HADOOP_MAPRED_HOME=/usr/local/hadoop
```

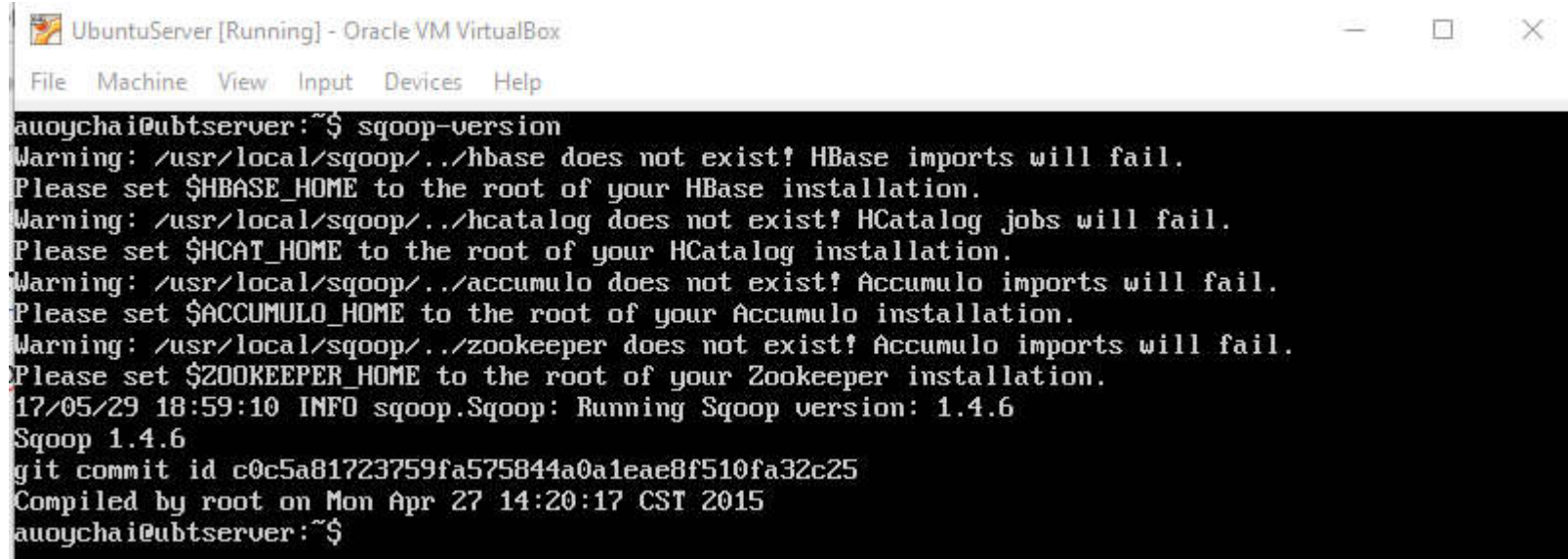
Prepare mysql connector for sqoop

```
$cp mysql-connector-java-2.0.14-bin.jar /usr/local/sqoop/lib
```

การ Start/Stop Apache Sqoop :

Verify Installation Completed :

\$sqoop-version

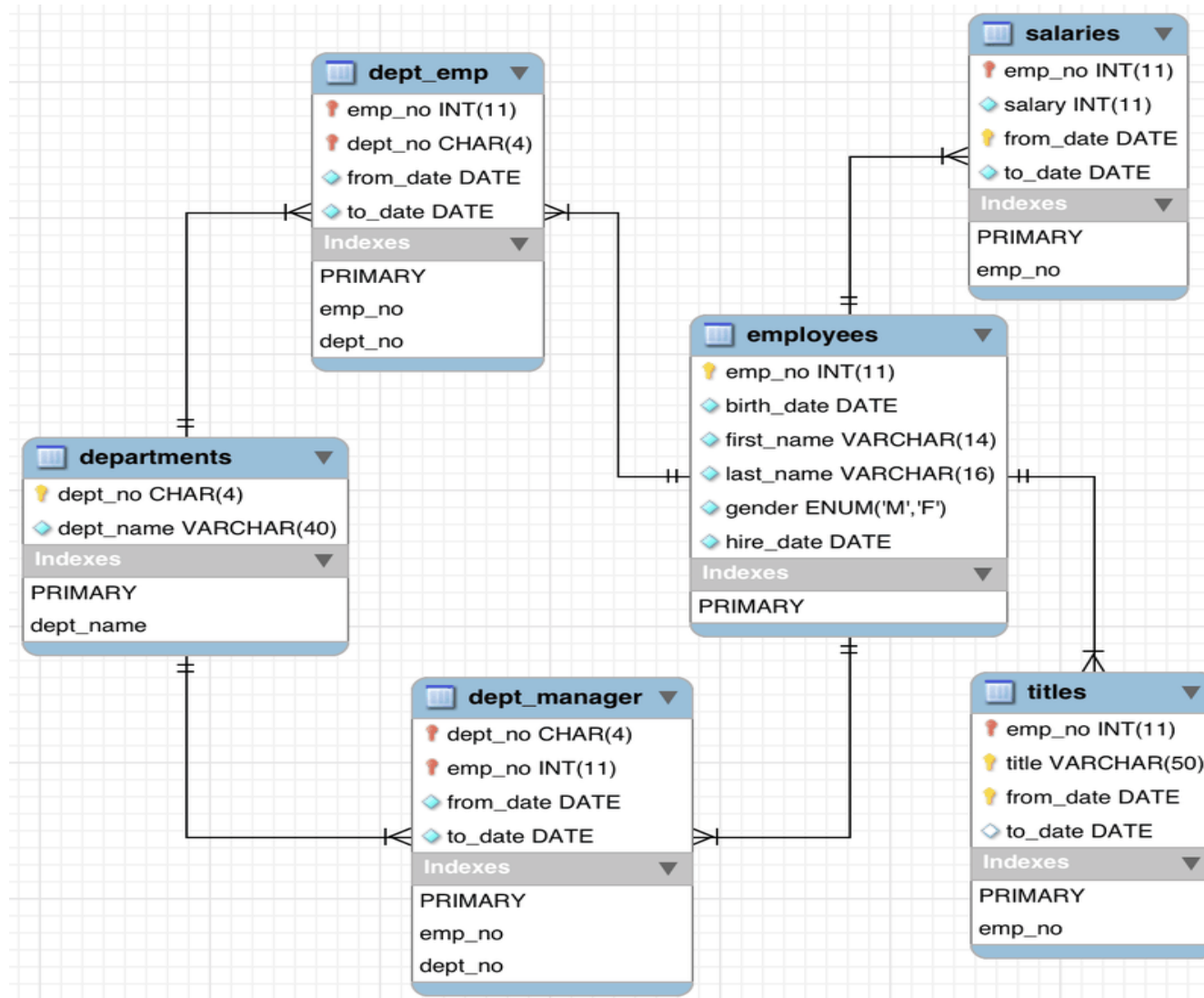


```
UbuntuServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
auoychai@ubtserver:~$ sqoop-version
Warning: /usr/local/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
17/05/29 18:59:10 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
Sqoop 1.4.6
git commit id c0c5a81723759fa575844a0a1eae8f510fa32c25
Compiled by root on Mon Apr 27 14:20:17 CST 2015
auoychai@ubtserver:~$
```

Pre-Requisite :

- 1). Hadoop started
- 2). Hive started
- 3). MySQL started & example db created

MySQL Example DB :



Import Data

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/<db-name>  
--username <username> --password <password> --table <table-name> --m  
<n> --warehouse-dir <hdfs-path>
```

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/employees  
--username auoychai --password 123456 --table employees --m 10 --  
warehouse-dir /user/lab/s2
```

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/employees  
--username auoychai --password 123456 --table employees --m 10 --target-  
dir /user/lab/s2
```

Import Data

- Condition:

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/employees
```

```
--username auoychai --password 123456
```

```
--query 'select * from employees where $conditions' --split-by emp_no --
```

```
m 10 --target-dir /user/lab/s2
```

Import Data

- Hive :

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/employees
```

```
--username auoychai --password 123456
```

```
--table employees --hive-import --hive-table emp.employees
```

```
--m 10
```

Export Data

- Hive :

```
$ sqoop import --connect jdbc:mysql://<mysql-server>/employees
```

```
--username auoychai --password 123456
```

```
--table employees --hive-import --hive-table emp.employees
```

```
--m 10
```


Hand-On :

Sqoop Job :

- Hive :

\$ -

The End

Big
data

Shift