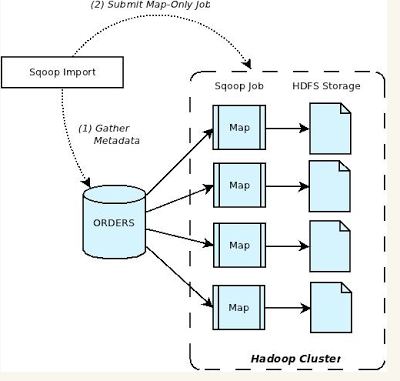
Apache sqoop

Apache Sqoop is a tool designed for efficiently transferring bulk data in a distributed manner between Apache Hadoop and structured datastores such as relational databases, enterprise data warehouses, and NoSQL systems.

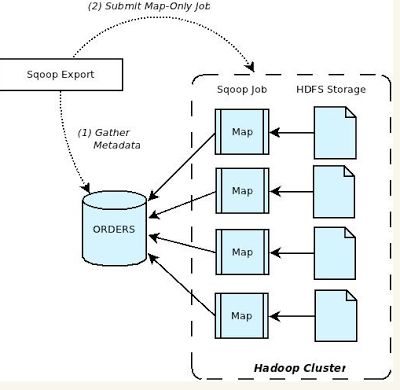
Sqoop can be used to import data into HBase, HDFS and Hive and out of it into RDBMS, in an automated fashion, leveraging Oozie for scheduling. It has a connector based architecture that supports plugins that provide connectivity to new external systems.

Behind the scenes, the dataset being transferred is split into partitions and map only jobs are launched for each partition with the mappers managing transferring the dataset assigned to it. Sqoop uses the database metadata to infer the types, and handles the data in a type safe manner.

The following diagrams are from the Apache documentation...  
  
**Import process:**

[](http://3.bp.blogspot.com/-LMAWjkPffak/UaylENqZAII/AAAAAAAAAHA/s2KHJlA6hsE/s1600/Screen+Shot+2013-06-03+at+9.15.20+AM.png)

**Export process:**

[](http://4.bp.blogspot.com/-SqVDLUfBIxQ/UayoHIzsUOI/AAAAAAAAAHY/0PnqlI8QuHU/s1600/Screen+Shot+2013-06-03+at+9.28.24+AM.png)

**Sqoop Installation :-**

**Untar the**  the sqoop.tar.gz to the home folder (/home/esak)

$untar <sqoop.tar.gz>

Copy the mysql-connector-java-5.1.27.jar to the /home/esak/sqoop/lib folder

If we want to use the oracle db we have use oracle specific jar and copy it to the above lib folder

1. To list Databases

Sqoop list-databases

--connect jdbc:mysql://<mysqlserver> /sqldbname –username root –password root

1. To list Tables

sqoop list-tables --connect jdbc:mysql://localhost/sqoopJpa -username root -password root

Import to HDFS

Sqoop import - - connect jdbc:mysql://<mysqlserver>/sqoopJpa

-username root –password root

- - table employee

- - target dir -/sqoop/output

- m 2

**Run the above command from the file**

**$ vim sqoopImport.txt**

Sqoop import - - connect jdbc:mysql://<mysqlserver>/sqoopJpa

-username root - -password root

To execute this file

$ sqoop - - options-file <sqoopImport.txt> - - table employee - -target dir /sqoop – m2

sqoop --options-file SqoopImportOptions.txt --table employee -m 1 --target-dir <file:///home/esak/JPA/Sqoop/out/employee_sqoopfile>

To execute a sql command from sqoop

$ Sqoop eval --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root -e "select \* from employee"

$ sqoop eval --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root -e "insert into employee values (1525,"esak","sankar",3456)";

To list all the Tables

$ sqoop import-all-tables --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root --warehouse-dir /sqoop/

Below parameter points to default hive table location.It can be used for dev purpose, where you just want to perform some tests on internal tables.

--warehouse-dir

Below parameter points to some hdfs location, where you can mount external hive tables.This is useful in production environment, where you want every data to be available to some external dir and external table.

--target-dir

--warehouse-dir

***generally you use this option when you're importing all the tables with import-all-tables tool using sqoop.*** This directory can be anything, either your hive /data/warehouse directory or some other parent directory. All the tables will be imported in this parent directory.

--target-dir

**This option is used when you've to import a single table using import-table tool.** For each table you've to mention the directory and it must not already exist in the path.

Sqoop code gen

The codegen tool generates Java classes which encapsulate and interpret imported records. The Java definition of a record is instantiated as part of the import process, but can also be performed separately. For example, if Java source is lost, it can be recreated. New versions of a class can be created which use different delimiters between fields, and so on.

**Import all rows of a table in my SQL, but specific columns of the table**

Import all columns with where clause

$ sqoop import --connect jdbs:mysql://<mysqlserver>/sqoopJpa --username root --password root --table employee --where "empno > 100" --as-textfile -m1 --target-dir /sqoop/output

sqoop import --connect jdbc:mysql://localhost/my --username user --password 1234 --query 'select \* from table where id=5 AND $CONDITIONS' --split-by table.id --target-dir /dir

**--split-by** is used to distribute the values from table across the mappers uniformly i.e. say u have 100 unique records(primary key) and if there are 4 mappers, --split-by (primary key column) will help to distribute you data-set evenly among the mappers.

**$CONDITIONS** is used by Sqoop process, it will replace with a unique condition expression internally to get the data-set. If you run a parallel import, the map tasks will execute your query with different values substituted in for $CONDITIONS. e.g., one mapper may execute "select bla from foo WHERE (id >=0 AND id < 10000)", and the next mapper may execute "select bla from foo WHERE (id >= 10000 AND id < 20000)" and so on.

Sqoop creates splits based on values in a particular column of the table which is specified by --split-by by the user through the import command. If it is not available, the primary key of the input table is used to create the splits.

**Reason to use :** Sometimes the primary key doesn't have an even distribution of values between the min and max values(which is used to create the splits if --split-by is not available). In such a situation you can specify some other column which has proper distribution of data to create splits for efficient imports.

**--boundary-query :** By default sqoop will use query **select min(), max() from**to find out boundaries for creating splits. In some cases this query is not the most optimal so you can specify any arbitrary query returning two numeric columns using --boundary-query argument.

**Reason to use :** If --split-by is not giving you the optimal performance you can use this to improve the performance further.

**Boundary Query**

By default sqoop will use query select min(<split-by>), max(<split-by>) from <table name> to find out boundaries for creating splits. In some cases this query is not the most optimal so you can specify any arbitrary query returning two numeric columns using --boundary-query argument.

**Fetch Size and Compression**

Sometimes we would need to keep imported data on HDFS in compressed state in order to reduce the overall diskutilization. As sqoop is based on Map Reduce execution, it inherits Hadoop's all compression features. It allows is to save imported data into various compressed formats e.g. GZIP or BZ2 etc. We can execute following query to use by default gzip compression,

sqoop import --connect:jdbc:mysql://<mysqlserver>/sqoopJpa --table employee --username root --password root -m 2 --compress –compress --target-dir /sqoop/output

This will store all files with .gz which is the default one

The output:

$ hadoop fs -ls -R sqoop-mysql/CompressedSample | grep part\*

-rw-r--r--   3 airawat airawat     896377 2013-05-31 23:49 sqoop-mysql/CompressedSample/part-m-00000.gz

sqoop import --connect:jdbc:mysql://<mysqlserver>/sqoopJpa --table employee --username root --password root -m 2 --compress --compress

--compression-codec org.apache.hadoop.io.compress.BZip2Codec

This will store all files with .bz2.

This argument specifies to sqoop the number of entries to read from database at once.

$ sqoop --options-file SqoopImportOptions.txt \

--query 'select EMP\_NO,FIRST\_NAME,LAST\_NAME from employees where $CONDITIONS' \

--fetch-size=50000 \

--split-by EMP\_NO \

--direct \

--target-dir /user/airawat/sqoop-mysql/FetchSize

Incremental Upload

Arguments

--check-column (col): Specifies the column to be examined when determining which rows to import.

--incremental (mode): Specifies how Sqoop determines which rows are new. Legal values for mode include append and lastmodified.

--last-value (value): Specifies the maximum value of the check column from the previous import

sqoop import --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root

--table employee

--check-column Empid

--incremental append

--last-value 1500

--target-dir /sqoop/output

sqoop import --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root

--query 'select EmpId , EmpName , LastName from employee where $conditions'

--check-column Empid

--incremental append

--last-value 1500

--split-by EmpId

--direct

--target-dir /sqoop/output

**Output Line Formatting In Sqoop**

# [How to specify fields delimiter when import MySQL into Hive with Sqoop?](http://stackoverflow.com/questions/38364936/how-to-specify-fields-delimiter-when-import-mysql-into-hive-with-sqoop)

You are using --hive-import, it will create hive table for you **IF NOT EXISTS**. It will create with Hive's default delimiter - fields terminated by : CTRL A and lines terminated by : \n

As per drill [docs](https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html#_importing_data_into_hive):

Even though Hive supports escaping characters, it does not handle escaping of new-line character.

Hive will have problems using Sqoop-imported data if your database’s rows contain string fields that have Hive’s default row delimiters (\n and \r characters) or column delimiters (\01characters) present in them. You can use the --hive-drop-import-delims option to drop those characters on import to give Hive-compatible text data. Alternatively, you can use the --hive-delims-replacement option to replace those characters with a user-defined string on import to give Hive-compatible text data.

You can simply use --hive-drop-import-delims in your query and it will drop \n.

sqoop import --connect jdbc:mysqll//localhost:3306/mysqldb \

--username user --password pwd --table mysqltbl \

--hive-import --hive-overwrite \

--hive-table hivedb.hivetbl -m 1 \

--hive-drop-import-delims \

--null-string '\\N' \

--null-non-string '\\N' \

If you want to replace your own string (say space i.e. " "), you can use --hive-delims-replacement.

sqoop import --connect jdbc:mysqll//localhost:3306/mysqldb \

--username user --password pwd --table mysqltbl \

--hive-import --hive-overwrite \

--hive-table hivedb.hivetbl -m 1 \

--hive-delims-replacement " " \

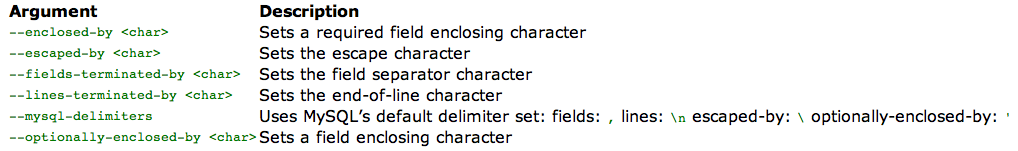
--null-string '\\N' \

--null-non-string '\\N' \

Hive

**Table 1.6. Sqoop Command Options for Importing Data into Hive**

| Sqoop Command Option | Description |
| --- | --- |
| --hive-home <directory> | Overrides $HIVE\_HOME. |
| --hive-import | Imports tables into Hive using Hive's default delimiters if none are explicitly set. |
| --hive-overwrite | Overwrites existing data in the Hive table. |
| --create-hive-table | Creates a hive table during the operation. If this option is set and the Hive table already exists, the job will fail. Set to falseby default. |
| --hive-table <table\_name> | Specifies the table name to use when importing data into Hive. |
| --hive-drop-import-delims | Drops the delimiters \n, \r, and \01 from string fields when importing data into Hive. |
| --hive-delims-replacement | Replaces the delimiters \n, \r, and \01 from strings fields with a user-defined string when importing data into Hive. |
| --hive-partition-key | Specifies the name of the Hive field on which a sharded database is partitioned. |
| --hive-partition-value <value> | A string value that specifies the partition key for data imported into Hive. |
| --map-column-hive <map> | Overrides the default mapping from SQL type to Hive type for configured columns |



### 2) Import data into Hive

2.1Basic Import

sqoop --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root

--table employee

-m 1

--hive-import

--create-hive-table

--hive-table hemp

--target-dir /sqooop

--fields-terminated-by ,

--enclosed-by ' \" '

--escaped-by \\

hive> select \* from hemp;

"d009" "Customer Service"

"d005" "Development"

"d002" "Finance"

"d003" "Human Resources"

"d001" "Marketing"

"d004" "Production"

"d006" "Quality Management"

"d008" "Research"

"d007" "Sales"

**Importing into Hive with partitions**

mysql> select gender, count(\*) from employees group by gender;    
+--------+----------+  
| gender | count(\*) |  
+--------+----------+  
| M      |   179973 |  
| F      |   120051 |

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

sqoop import --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root

--query 'select EMP\_NO,birth\_date,first\_name,last\_name,hire\_date from employee where gender = "M" AND $CONDITIONS '

--split-by empId

--m 6

--hive-import

--create-hive-table

--hive-table hivenewparttable\

--hive-partition-key gender

--hive-partition-value 'M'

--fields-terminated-by ,

--enclosed-by '\"'

--escaped-by \\

For Gender F

sqoop import --connect jdbc:mysql://<mysqlserver>/sqoopJpa --username root --password root

--query 'select EMP\_NO,birth\_date,first\_name,last\_name,hire\_date from employee where gender = "M" AND $CONDITIONS '

--split-by EmpId

-m 6

--hive-import

--hive-overwrite

--hive-table hnewpartf

--hive-partition-key gender

--hive-partition-value 'F'

--fields-terminated-by ,

--enclosed-by '\"'

--escaped-by \\

--target-dir /user/esak

Note :

1) Gender Column should not be inclueded in the select Statement

2) --hive-partition-key gender {column name} and --hive-partition-value 'M'

3) if we want to overwrite the daa which is already available in hive --hive-overwrite

Dynamic Partitioning

Currently Sqoop Doesn’t Support Dynamic Partitioning .

**HOW DO I**  
I have external databases which contain tables with complex data types, like VARCHAR(32) and LONG. I would like to import these tables using Sqoop. When I used standard import commands to create hive tables, all the columns were strings and integers. I would like to automatically create tables with columns that have the same specifications as my original tables. How can I do this using Sqoop  
   
**STEPS**  
Through the use of HCatalog, it is possible for Sqoop to create and import complex data types automatically. A sample command line that uses a hive table as the source data, and creates a hive table with the same column specifications automatically is shown below.

1. user@host > sqoop import \
2. --connect jdbc:mysql://mysqlhost.example.com/mysql\_database \
3. --username user\_name -P \
4. --table mysql\_tablename \
5. --hcatalog-database hcat\_hive\_db --hcatalog-table hcat\_hive\_table --create-hcatalog-table \
6. --driver com.mysql.jdbc.Driver \
7. -m 1

   
**BACKGROUND/DISCUSSION**  
As written, the Sqoop import with the various hive options will generally "flatten" the incoming data types into three: string, int, and boolean. There are a few other types, but most incoming data types map to these three. It requires significant extra work to specify the target column mapping and length using hive import semantics. This can be tedious and error prone for tables with large numbers of columns, or when importing many tables, or both.   
  
Using Sqoop with HCatalog semantics resolves this problem, since HCatalog has a better understanding of data types, and will create or import column data types in a much more granular fashion. It will make a best effort to match types as closely as possible, using more granular DDL and select statements as needed. It cannot match all data types, as it depends on the underlying function of HCatalog and Hive SQL, but it will make a best effort. For typical data types such as floats, VARCHAR, strings, and the like, use of HCatalog functions in the Sqoop import will result in Hive tables that are much closer to those of the original RDBMS tables.

Export

### Apache Sqoop documentation on the "export" tool

Exports are performed by multiple writers in parallel. Each writer uses a separate connection to the database; these have separate transactions from one another. Sqoop uses the multi-row INSERT syntax to insert up to 100 records per statement. Every 100 statements, the current transaction within a writer task is committed, causing a commit every 10,000 rows. This ensures that transaction buffers do not grow without bound, and cause out-of-memory conditions. Therefore, an export is not an atomic process. Partial results from the export will become visible before the export is complete.

Exports may fail for a number of reasons:

* Loss of connectivity from the Hadoop cluster to the database (either due to hardware fault, or server software crashes)
* Attempting to INSERT a row which violates a consistency constraint (for example, inserting a duplicate primary key value)
* Attempting to parse an incomplete or malformed record from the HDFS source data
* Attempting to parse records using incorrect delimiters
* Capacity issues (such as insufficient RAM or disk space)

If an export map task fails due to these or other reasons, it will cause the export job to fail. The results of a failed export are undefined. Each export map task operates in a separate transaction. Furthermore, individual map tasks commit their current transaction periodically. If a task fails, the current transaction will be rolled back. Any previously-committed transactions will remain durable in the database, leading to a partially-complete export.

**Basic Export**

sqoop export --connect jdbc:mysql//<mysqlserver>/SqoopJpa --username root --password root

--table employee\_export

--staging-table employee\_stg

--clear-staging-table

-m 1

--export-dir /usr/warehousw

Note: Even without the clear staging table argument, I found that the staging table was empty, however,

the command output clearly indicates that the staging table was used.

.....

....

13/06/04 09:54:18 INFO manager.SqlManager: Migrated 300024 records from `employees\_exp\_stg` to `employees\_export`

Export Using Insert Mode

sqoop export --connect jdbc:mysql//<mysqlserver>/sqoopJpa --username root --password root

--table employee\_export

-m 2

--export-dir /user/esak/warehouse

--update-key Empid

--update-mode allowinsert

--export-dir /usr/esak

Another Detailed Example for insert/Upinsert

Update data from HDFS into a table in a relational database

Create emp table tbl in mysql test db

create table emp

(

id int not null primary key,

name varchar(50)

);

vi emp --> create file with below contents

1,Thiru

2,Vikram

3,Brij

4,Sugesh

Move the file to hdfs

hadoop fs -put emp <dir>

Execute the below sqoop job to export the data to the mysql

sqoop export --connect <jdbc connection> \

--username sqoop \

--password sqoop \

--table emp \

--export-dir <dir> \

--input-fields-terminated-by ',';

Verify the data in the mysql table

mysql> select \* from emp;

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

| id | name |

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

| 1 | Thiru |

| 2 | Vikram |

| 3 | Brij |

| 4 | Sugesh |

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

update the emp file & move the updated file into hdfs. contents of the updated file

1,Thiru

2,Vikram

3,Sugesh

4,Brij

5,Sagar

Sqoop export for upsert - Update if the key matches else insert.

sqoop export --connect <jdbc connection> \

--username sqoop \

--password sqoop \

--table emp \

--update-mode allowinsert \

--update-key id \

--export-dir <dir> \

--input-fields-terminated-by ',';

Note: --update-mode <mode> - we can pass two arguments "updateonly" - to update the records. this will update the records if the update key matches.

if you want to do upsert (If exists UPDATE else INSERT) then use "allowinsert" mode.

example:

--update-mode updateonly \ --> for updates

--update-mode allowinsert \ --> for upsert

verify the results:

mysql> select \* from emp;

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

| id | name |

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

| 1 | Thiru |

| 2 | Vikram |

| 3 | Sugesh |--> Previous value "Brij"

| 4 | Brij |--> Previous value "Sugesh"

| 5 | Sagar |--> new value inserted

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

Export Hive table back to an RDBMS:

By default, Hive will stored data using ^A as a field delimiter and \n as a row delimiter.

$ bin/sqoop export --connect jdbc:mysql://localhost/test\_db --table tableName  --export-dir /user/hive/warehouse/tableName --username root --password password -m 1 --input-fields-terminated-by '\001'  
  
**where** '\001' is octal representation of ^A

sqoop export --connect jdbc:mysql://localhost/test --table CONTACT --export-dir /apps/hive/warehouse/contact\_hive

sqoop export \  
--connect jdbc:mysql://airawat-mysqlserver-node/employees \

--username myUID \

--password myPWD \

--table departments\_export\_hive  \

--direct \  
--enclosed-by '\"' \

--export-dir /user/hive/warehouse/departments\_mysql  
.  
.  
.  
13/06/04 11:25:27 INFO mapreduce.ExportJobBase: Transferred 1.0869 KB in 69.1858 seconds (16.0871 bytes/sec)  
13/06/04 11:25:27 INFO mapreduce.ExportJobBase: Exported 9 records.

Exporting Partition Table

Note 1: With Sqoop 1.4.2., we need to issue a sqoop statement for every partition individually.    
Note 2:  In the export, the partition key will not be inserted, you have to issue an update statement for the same.

sqoop export \  
--connect jdbc:mysql://airawat-mysqlserver-node/employees \  
--username myUID \  
--password myPWD \  
--table employees\_export\_hive  \  
--direct \  
--enclosed-by '\"' \  
--export-dir /user/hive/warehouse/employees\_import\_parts/gender=M

**Execute partition key update:**  
  
mysql> update employees\_export\_hive set gender='M' where (gender="" or gender is null);  
  
Query OK, 179973 rows affected (1.01 sec)

**Export partition where gender is F:**  
  
$ sqoop export \  
--connect jdbc:mysql://airawat-mysqlserver-node/employees \  
--username myUID \  
--password myPWD \  
--table employees\_export\_hive  \  
--direct \  
--enclosed-by '\"' \  
--export-dir /user/hive/warehouse/employees\_import\_parts/gender=F

**Execute partition key update:**  
  
mysql> update employees\_export\_hive set gender='F' where (gender="" or gender is null);  
  
Query OK, 120051 rows affected (1.02 sec)

### Exporting out of Hive into mysql in update mode

#### C1. Prep work

#### C1.1. Issue the following update in mysql to the department table to try update functionality

mysql> update departments\_export\_hive set dept\_name="Procrastrinating" where dept\_no="d001";

Query OK, 1 row affected (0.00 sec)

Rows matched: 1  Changed: 1  Warnings: 0

mysql> select \* from departments\_export\_hive;

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

| dept\_no | dept\_name          |

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

| d002    | Finance            |

| d003    | Human Resources    |

| d001    | Procrastrinating   |

| d008    | Research           |

| d007    | Sales              |

| d009    | Customer Service   |

| d005    | Development        |

| d004    | Production         |

| d006    | Quality Management |

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

9 rows in set (0.00 sec)

#### C2. Sqoop export command:

$ sqoop export \

--connect jdbc:mysql://airawat-mysqlserver-node/employees \

--username myUID \

--password myPWD \

--table departments\_export\_hive  \

--enclosed-by '\"' \

--update-key "dept\_no" \

--update-mode updateonly \

--export-dir /user/hive/warehouse/departments\_mysql

#### C3. Results:

mysql> select \* from departments\_export\_hive;

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

| dept\_no | dept\_name          |

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

| d002    | Finance            |

| d003    | Human Resources    |

| d001    | Marketing          |

| d008    | Research           |

| d007    | Sales              |

| d009    | Customer Service   |

| d005    | Development        |

| d004    | Production         |

| d006    | Quality Management |

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

9 rows in set (0.00 sec)

# Can I sqoop data from Oracle into a partitioned Hive/Hcat table?

Yes. If the table already exists and has a partition, you can use the syntax in the example below.  First, create the oracle table and load some test data:

1. sqlplus sqoop/sqoop@XE
2. create table test (col1 integer not null primary key, col2 char(10));
3. insert into test (col1, col2) select rownum, to\_char(rownum) from dual connect by level <= 100;

Now sqoop into your hive table, specifying the partition value:

1. Hive> describe TEST\_P;
2. col1 double
3. col2 string
4. country string
6. # Partition Information
7. # col\_name data\_type comment
8. country string
10. sqoop import --connect "jdbc:oracle:thin:@localhost:1521:XE" --password "sqoop" --username "sqoop" --table TEST --hive-import --create-hive-table --hive-table "TEST\_P" --hive-partition-key country --hive-partition-value "USA"

Note: HIVE only allows a single partition column. For the more complicated case you can use hcatalog as follows: 

1. hcat -e "create table TEST\_HC (col1 bigint, col2 char(10)) partitioned by (country string, state string);"
3. sqoop import --connect "jdbc:oracle:thin:@localhost:1521:XE" --password "sqoop" --username "sqoop" --table TEST
4. --hcatalog-table "TEST\_HC" --hcatalog-partition-keys country,state --hcatalog-partition-values "USA","California"

### ****I have around 300 tables in a database. I want to import all the tables from the database except the tables named Table298, Table 123, and Table299. How can I do this without having to import the tables one by one?****

This can be accomplished using the import-all-tables import command in Sqoop and by specifying the exclude-tables option with it as follows-

sqoop import-all-tables

--connect –username –password --exclude-tables Table298, Table 123, Table 299

List all the columns

Sqoop import --m 1 --connect 'jdbc: sqlserver: //nameofmyserver; database=nameofmydatabase; username=DeZyre; password=mypassword' --query "SELECT column\_name, DATA\_TYPE FROM INFORMATION\_SCHEMA.Columns WHERE table\_name='mytableofinterest' AND \$CONDITIONS" --target-dir 'mytableofinterest\_column\_name'

Hadoop MapReduce cluster is configured to run a maximum of 4 parallel MapReduce tasks and the sqoop import can be configured with number of parallel tasks less than or equal to 4 but not more than 4.

### Resolved Issue / Error in sqoop 1.4

I've posted some issue in sqoop2  [here.](http://solaimurugan.blogspot.in/2013/07/error-working-with-sqoop-1992.html" \t "_blank)   
  
I had a good time when working with sqoop 1.4 connecting to PostgreSQL  database. I have faced few issues & managed to overcome those issues.  
  
**ERROR 1)**

....

....

sqoop.Sqoop: Got exception running Sqoop: java.lang.RuntimeException:

Could not load db driver class: com.mysql.jdbc.Driver

java.lang.RuntimeException: Could not load db driver class: com.mysql.jdbc.Driver

**SOLUTION**  
[download](http://dev.mysql.com/downloads/connector/) & put MySQL connector to the $SQOOP\_HOME/lib dir.

$SQOOP\_HOME/lib

**ERROR 2)**

....

....

sqoop.Sqoop: Got exception running Sqoop: java.lang.RuntimeException: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

.....

Caused by: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

.....

Caused by: java.net.ConnectException: Connection refused

**SOLUTION**  
check the status of the MySQL up & running.ensure user has rights to access the database.for more [find here](http://sqoop.apache.org/docs/1.4.3/SqoopUserGuide.html#_mysql_connection_failure).  
  
step 1)  verify that you can connect to the database from the node where you are running Sqoop:

root@boss[~]#mysql --host=localhost --database=test --user=root --password=root

step 2) Grant permissions to the user to access the database

root@boss[~]#mysql --host=localhost --user=root --password=root

mysql>grant all privileges on test.\* to 'sqoopuser'@'%' identified by 'sqooppassword'

**ERROR 3)**

....

....

orm.ClassWriter: No parser available for Java type org.apache.hadoop.io.BytesWritable

**SOLUTION**   
  in postgresql/oracle , column with data type bytea/binary or varbinary  contains null value means through above error. 

update the "bytea/binary or varbinary" column with not-null value.

**ERROR 4)**

....

....

INFO ipc.Client: Retrying connect to server: /127.0.0.1:8021. Already tried 0 time(s).

ERROR tool.ImportTool: Encountered IOException running import job: java.net.ConnectException:

Call to /127.0.0.1:8021 failed on connection exception: java.net.ConnectException: Connection refused

**ERROR 5)**

....

....

ERROR tool.ImportTool: Encountered IOException running import job:

java.io.IOException: Call to /127.0.0.1:8021 failed on local exception:

java.io.IOException: Connection reset by peer

**SOLUTION (4,5)**  
check the status of hadoop process

root@boss[~]#jps

20668 JobTracker

20434 DataNode

20299 NameNode

20569 SecondaryNameNode

20799 TaskTracker20844 Jps

ensure all the 5 process are running successfully. to avoid the error 5) JobTracker must be up & running.   
  
  
**ERROR 6)**

....

....

localhost: ssh: Could not resolve hostname localhost: Name or service not known

**ERROR 7)**

....

....

ERROR org.apache.hadoop.hbase.master.HMasterCommandLine: Failed to start master

java.net.ConnectException:

Call to localhost/127.0.0.1:9000 failed on connection exception:

java.net.ConnectException: Connection refused

**SOLUTION (6,7)**  
I've faced said error while running hbase, to resolve the error, open & edit  /etc/hosts

ip/domain

127.0.0.1    localhost.localdomain

127.0.0.1    boss

127.0.0.1    hbase-master:q

 to resolve the error 7) 

127.0.0.1    hbase-master:q

must be added in file /etc/hosts   
  
I've connected PostgreSQL database using sqoop, tried out store the data across HDFS and analysis using hbase.