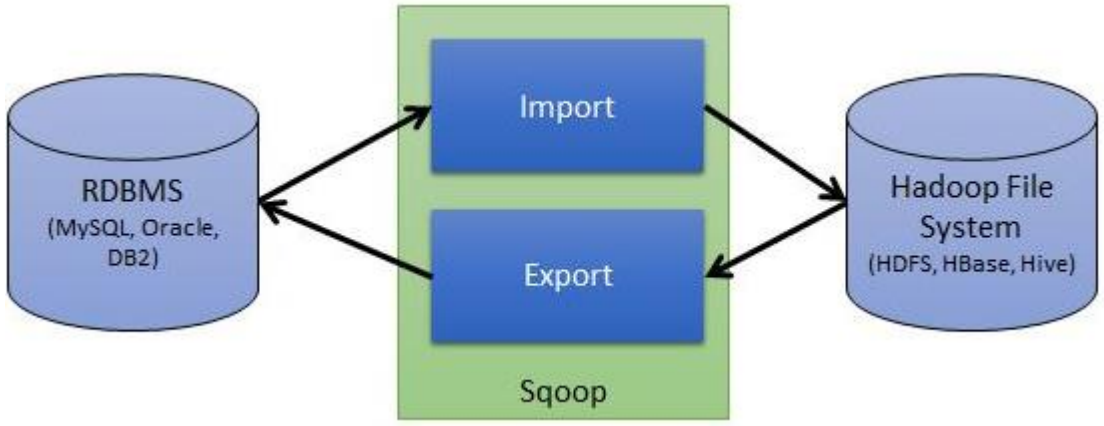
Apache Sqoop

What is Sqoop

- **Sqoop:** “SQL to Hadoop and Hadoop to SQL”

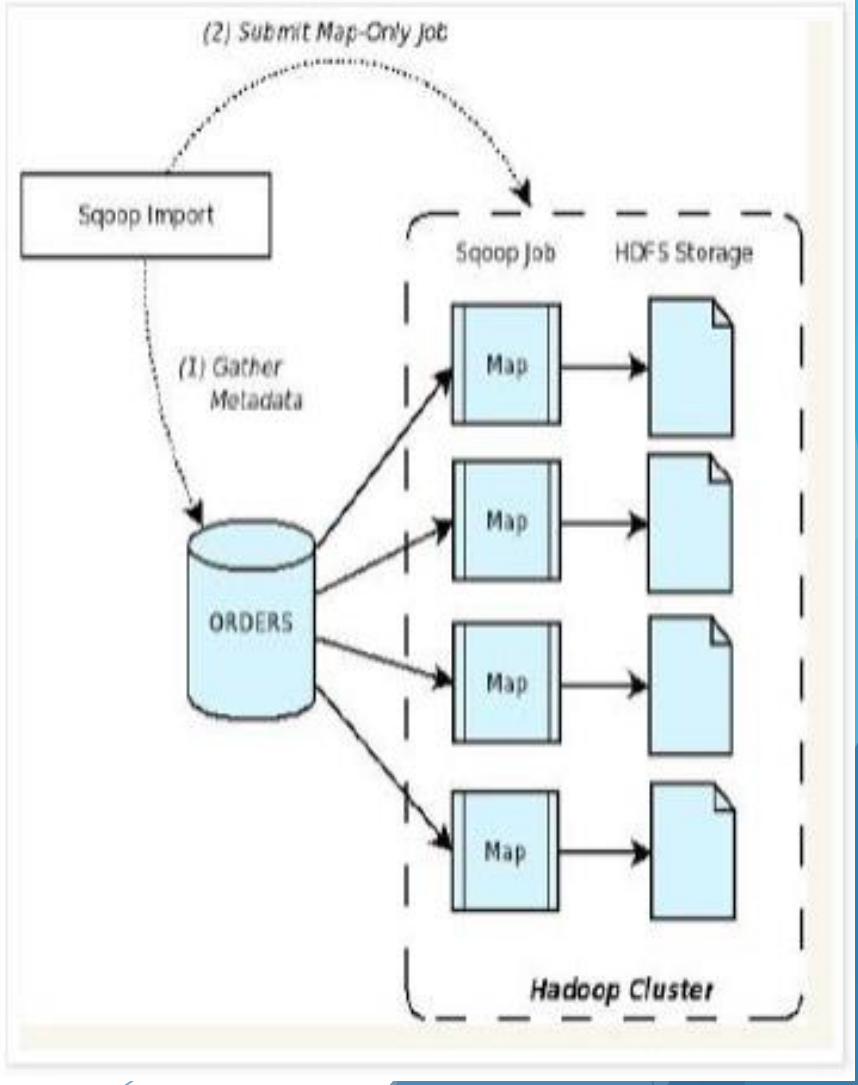
- Sqoop is a tool designed to transfer data between Hadoop and relational database servers.
- It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.
- It is provided by the Apache Software Foundation.



Sqoop Import

- The input to the import process is a database table
- Sqoop will read the table row-by-row into HDFS.
- The output of this import process is a set of files containing a copy of the imported table.

- The import process is performed in parallel. For this reason, the output will be in multiple files.
- These files may be delimited text files (for example, with commas or tabs separating each field) or binary Avro or SequenceFiles



Import Options

```
sqoop import
 --connect jdbc:mysql://host/nyse
 --table StockPrices
 --target-dir /data/stockprice/
 --as-textfile
```

```
sqoop import-all-tables \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop
```

```
sqoop import-all-tables \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --exclude-tables cities,countries
```

Freeform SQL

- Instead of using table import, use free-form query import.
- Use the --query argument to specify which rows to select from a table.
- Sqoop will not use the database catalog to fetch the metadata

```
sqoop import
 --connect jdbc:mysql://host/nyse
 --query "SELECT * FROM StockPrices s
 WHERE s.Volume >= 1000000
 AND \$CONDITIONS"
 --target-dir /data/highvolume/
 --as-textfile
 --split-by StockSymbol
```

- --split-by parameter with the column is used for slicing the data into multiple parallel tasks.
- Using --query is limited to simple queries

Sqoop Import - Arguments

- To enter a password for the data source on the command line, use the `-P` option in the connection string.
- To specify a file where the password is stored, use the `--password-file` option.

Password on command line:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
<data to import> \
--username <username> \
-P
```

Specify password file:

```
sqoop import --connect jdbc:mysql://db.foo.com:3306/bar \
--table EMPLOYEES \
--username <username> \
--password-file ${user.home}/.password
```

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

The `--null-string` and `--null-non-string` arguments are optional.\ If not specified, then the string "null" will be used.

Sqoop with Hive

- Sqoop can import your data directly into Hive.
- Add the parameter --hive-import to your command to enable it

- Example

$ sqoop create-hive-table
  --connect jdbc:mysql://localhost:3306/flights
  --table Flights
  --username root
  --password cloudera

Argument	Description
--hive-home <dir>	Override $HIVE_HOME
--hive-import	Import tables into Hive (Uses Hive's default delimiters if none are set.)
--hive-overwrite	Overwrite existing data in the Hive table.
--create-hive-table	If set, then the job will fail if the target hive table exits. By default this property is false.
--hive-table <table-name>	Sets the table name to use when importing to Hive.
--hive-drop-import-delims	Drops \n, \r, and \01 from string fields when importing to Hive.
--hive-delims-replacement	Replace \n, \r, and \01 from string fields with user defined string when importing to Hive.
--hive-partition-key	Name of a hive field to partition are sharded on
--hive-partition-value <v>	String-value that serves as partition key for this imported into hive in this job.

sqoop import -m 1 --connect jdbc:mysql://localhost:3306/flights --table FLIGHTS --username root --password cloudera
--hive-import

Sqoop with HBase

To enable import into HBase, there are two additional parameters:

- **--hbase-table**

Specifies the name of the table in HBase to which you want to import your data.

- **--column-family**

Specifies into which column family Sqoop will import your table's data.

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table cities \
  --hbase-table cities \
  --column-family world
```

Sqoop Jobs

- The Sqoop metastore allows you to retain your job definitions and to easily run them anytime.
- Each saved job has a logical name that is used for referencing.
- To create a sqoop job:

```
sqoop job \
  --create visits \
  -- \
  import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table visits \
  --incremental append \
  --check-column id \
  --last-value 0
```

- List all retained jobs using the --list parameter

sqoop job –list

- View content of the saved job definitions using the –show parameter

sqoop job --show visits

- Remove the old job definitions that are no longer needed with the –delete parameter

sqoop job --delete visits

Incremental Import

- Sqoop provides an incremental import mode which can be used to retrieve only rows newer than some previously-imported set of rows.

Sqoop supports two types of incremental imports:

- append

For importing the newly created rows to the existing data

- lastmodified

Used when existing rows needs to be updated

- --incremental argument is used for incremental import

Incremental Import – Append

- Incremental import in append mode will allow you to transfer only the newly created rows.

Incremental import also requires two additional parameters:

- --check-column indicates a column name that should be checked for newly appended data
- -last-value contains the last value that successfully imported into Hadoop.

- Sqoop when running in incremental mode, always prints out the value of the last imported row.

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table visits \
  --incremental append \
  --check-column id \
  --last-value 1
```

Incremental Import – Lastmodified

- This is used for mutable data. The data which is getting changed
- Use the lastmodified mode instead of the append mode.
- The incremental mode lastmodified requires a column holding a date value (suitable types are date, time, datetime, and timestamp) containing information as to when each row was last updated.

```
sqoop import \
  --connect jdbc:mysql://mysql.example.com/sqoop \
  --username sqoop \
  --password sqoop \
  --table visits \
  --incremental lastmodified \
  --check-column last_update_date \
  --last-value "2013-05-22 01:01:01"
```

Sqoop - Merge

- The merge tool allows you to combine two datasets where entries in one dataset should overwrite entries of an older dataset
- For ex: An incremental import run in last-modified mode will generate multiple datasets in HDFS where successively newer data appears in each dataset.
- The merge tool will "flatten" two datasets into one, taking the newest available records for each primary key.
- When merging the datasets, it is assumed that there is a unique primary key value in each record.
- The column for the primary key is specified with --merge-key
- It can be used in conjunction with Sqoop import

```
$ sqoop merge --new-data newer --onto older --target-dir merged
    --jar-file datatypes.jar --class-name Foo --merge-key id
```

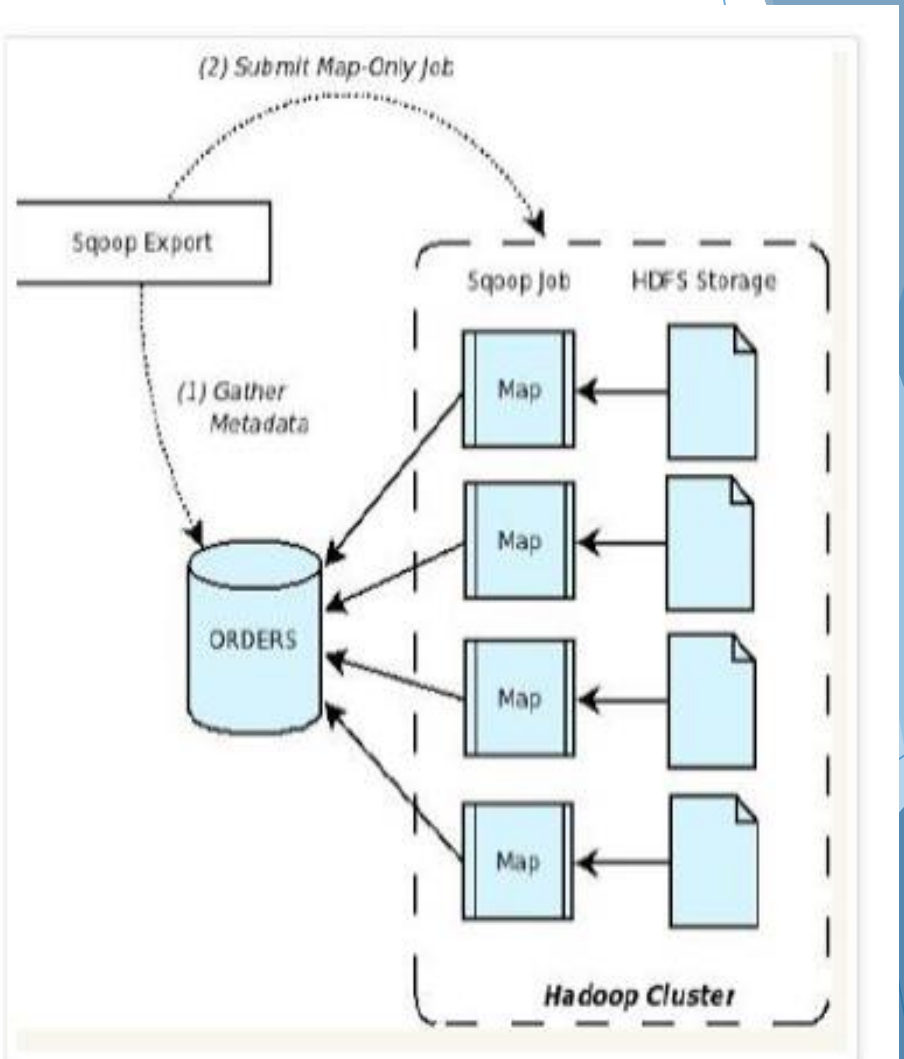
sqoop import -m 1 --connect jdbc:mysql://localhost:3306/mysql --table city_date --username xxxx --password xxxxx --incremental lastmodified --check-column mdate --last-value '2018-08-16 00:08:56.0' --target-dir /user/cloudera/city_date --merge-key sid

Sqoop Export

- After manipulating the imported records (for example, with MapReduce or Hive) you may have a result data set which you can then export back to the relational database.

- The export tool exports a set of files from HDFS back to an RDBMS.
- The files given as input to Sqoop contain records, which are called as rows in table.

- Those are read in parallel and parsed into a set of records and inserted them as new rows in a target database table



Sqoop Export Options

- The target table must already exist in the database
- The input files are read and parsed into a set of records according to the user-specified delimiters.
- The default operation is to transform these into a set of INSERT statements that inject the records into the database.
- By default, sqoop-export appends new rows to a table
- In "update mode," Sqoop will generate UPDATE statements that replace existing records in the database,
- Each input record is treated as an UPDATE statement that modifies an existing row
- In "call mode" Sqoop will make a stored procedure call for each record.

```
sqoop export
 --connect jdbc:mysql://host/mylogs
 --table LogData
 --export-dir /data/logfiles/
 --input-fields-terminated-by "\t"
```

Troubleshooting

Known Issues:

- MySQL connection failure
  - Verify permissions in the file system
- Connection Reset Errors
- Argument mistakes
- Case-Sensitive Catalog Query Errors
- SQL command not properly ended

Solution Approach:

- Check the version of JDBC connector
- Understand table schema
- Check logs for any issue
- Validate error messages

Thank You

Keerthiga Barathan