SQOOP IMPORT

TO LIST DATABASES IN HADOOP USING SQOOP

```
sqoop-list-databases \
--connect "jdbc:mysql://quickstart.cloudera:3306" \
--username retail_dba \
--password cloudera
```

TO LIST TABLES IN HADOOP USING SQOOP

```
sqoop-list-tables \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username retail_dba \
--password cloudera
```

DISPLAYING TABLES USING SQOOP EVAL

```
sqoop-eval \
--connect "jdbc:mysql://10.0.2.15:3306" \
--username retail_dba \
--password cloudera \
--query "select * from retail_db.customers limit 10"
```

OR (-e)

```
sqoop-eval \
--connect "jdbc:mysql://10.0.2.15:3306" \
--username retail_dba \
--password cloudera \
-e "select * from retail_db.customers limit 10"
```

TO GET THE IP ADDRESS

```
ifconfig
10.0.2.15
```

```
sqoop-eval \
--connect "jdbc:mysql://10.0.2.15:3306" \
--username retail_dba \
--password cloudera \
--query "describe retail_db.orders"
```

SQOOP IMPORT (WHEN THERE IS A PRIMARY KEY)

1.USING TARGET DIR.

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--target-dir /queryresult
```

```
hadoop fs -ls /queryresult
hadoop fs -cat /queryresult/*
```

2.USING WAREHOUSE DIR.

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /ordersresult2
```

```
hadoop fs -ls /ordersresult2/orders
```

SQOOP IMPORT (WHEN THERE IS NO PRIMARY KEY)

1. MAKING THE NUMBER OF MAPPER 1

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/trendytech \
--username root \
--password cloudera \
--table people \
-m 1 \
--target-dir /peopleresult
```

2. SPLIT BY

1. ON NUMERIC COLUMN

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders_no_primarykey \
--split-by order_id \
--warehouse-dir /ordersresult2
```

2. TEXTUAL COLUMN (not recommended)

```
sqoop import \
-Dorg.apache.sqoop.splitter.allow_text_splitter=true \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders_no_primarykey \
--split-by "category_name" \
--warehouse-dir /ordersresult2
```

-D = Indicates start of the property
performance is not good

SQOOP AUTORESET TO ONE MAPPER (EX-1)

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders_no_primarykey \
--warehouse-dir /user/cloudera/npkresult
--autoreset-to-one-mapper \
-m 8 \
```

please autoreset to one mapper when there is no primary key
If there is primary key use number of mappers specified.

SQOOP AUTORESET TO ONE MAPPER (EX-2)

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--as-sequencefile \
--autoreset-to-one-mapper \
-m 4 \
--warehouse-dir /user/cloudera/sqoopdir
```

please autoreset to one mapper for all tables when there is no primary key
If there is primary key use number of mappers specified.

TO IMPORT ALL TABLES FROM DATABASE TO HDFS

```
sqoop import-all-tables \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--as-sequencefile \
-m 4 \
--warehouse-dir /user/cloudera/sqoopdir
```

SQOOP HELP

```
sqoop help
```

SQOOP VERSION

```
sqoop version
```

PARAMETERS WHICH CAN BE USED WITH COMMANDS

```
sqoop help eval
sqoop help import
```

ALTERNATIVE FOR PASSWORD

-P

```
sqoop-list-databases \
--connect "jdbc:mysql://quickstart.cloudera:3306" \
--username retail_dba \
-P
```

It will ask for the password while executing.

REDIRECTING LOGS

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /ordersresult2 1>query.out 2>query.err
```

query.out = All the output messages/records
query.err = All the errors or logs

```
cat query.out
cat query.err
```

COMPRESSION TECHNIQUES

1. USING GZIP

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--compress \
--warehouse-dir /user/cloudera/compresult
```

OR

```
--compress or -z
```

```
hadoop fs -ls /user/cloudera/compresult/orders
```
.gz extension

2.USING BZip2Codec

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--compression-codec BZip2Codec \
--warehouse-dir /user/cloudera/bzipcomp
```

```
hadoop fs -ls /user/cloudera/bzicomp/orders
```
.bz2 extension

IMPORT DATA WITH SELECTED COLUMN

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table customers \
--columns customer_id,customer_fname,customer_city \
--warehouse-dir /user/cloudera/customerresult
```

```
hadoop fs -ls /user/cloudera/customerresult/customers
hadoop fs -cat /user/cloudera/customerresult/customers/*
hadoop fs -cat /user/cloudera/customerresult/customers/part-m-00000 | head
```

WHERE CLAUSE

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--columns order_id,order_customer_id,order_status \
--where "order_status in ('complete ','closed')" \
--warehouse-dir /user/cloudera/result199
```

to count the number of lines = hadoop fs -cat /user/cloudera/result199/orders/* | wc -l

BOUNDARY QUERY CUSTOMIZATION

PASSING THE HARDCODED MIN AND MAX VALUES

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--boundary-query "SELECT 1,68883" \
--warehouse-dir /user/cloudera/boundaryq
```

USING NON PRIMARY KEY COLUMN

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table order_items \
--boundary-query "SELECT min(order_item_order_id),max(order_item_order_id) FROM
order_items WHERE order_item_order_id > 10000" \
--warehouse-dir /user/cloudera/boundaryq1
```

USING WHERE CLAUSE

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--columns order_id,order_customer_id,order_status \
--where "order_status in ('processing')" \
--warehouse-dir /user/cloudera/result8
```

DELIMITERS

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--fields-terminated-by '|' \
--lines-terminated-by ';' \
--target-dir /user/cloudera/result1234
```

CREATE HIVE TABLE SAME AS MYSQL TABLE WITH SAME SCHEMA

```
sqoop create-hive-table \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
-hive-table emps \
```

SQOOP VERBOSE

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--verbose \
--target-dir /queryresult8
```

verbose will generate more logs and debugging information.

SQOOP APPEND

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--target-dir /queryresult8 \
--append
```

append argument will append data to an exsiting dataset in HDFS.

DELETE TARGET DIRECTORY IF EXISTS

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--target-dir /queryresult8 \
--delete-target-dir
```

DEALING WITH NULLS WHILE IMPORTING DATA

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username retail_dba \
--password cloudera \
--table orders \
--target-dir /user/cloudera/result1234 \
--delete-target-dir \
--null-non-string "-1"
```

SQOOP EXPORT

HDFS to RDBMS

Target table must already exist in the database.

If file is present in local which you want to export in RDBMS then we first need to move that file into HDFS.

Suppose,we have CARD_TRANS.CSV file on desktop locally in cloudera then move it from LOCAL to HDFS

hadoop fs -mkdir /data =create directory in HDFS

hadoop fs -put Desktop/card_trans.csv /data = Local to Hdfs

hadoop fs -ls /data = check

```
CREATE TABLE card_transactions (
card_id BIGINT,
member_id BIGINT,
amount INT (10),
postcode INT (10),
pos_id BIGINT,
transaction_dt varchar (255),
status varchar (255),
PRIMARY KEY (card_id, transaction_dt));
```

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/banking \
--username root \
--password cloudera \
--table card_transactions \
--export-dir /data/card_trans-200913-220429.xlsx \
--fields-terminated-by ","
```

1. If the Job Fails? How to see what is gone wrong?

Go to logs --> Click on URL --> Click on number (1) of failed maps
--> Logs --> Here you will see the complete details.

2. How to make sure target table is not impacted?

<u>STAGING TABLE</u>

There should not be any partial transfer of data either full transfer or no transfer
So,Here comes the STAGING TABLE
Create staging table in our MYSQL
Schema of this staging table should be exactly like target table.

```
CREATE TABLE card_transactions_stage (
card_id BIGINT,
member_id BIGINT,
amount INT (10),
postcode INT (10),
pos_id BIGINT,
transaction_dt varchar (255),
status varchar (255),
PRIMARY KEY (card_id, transaction_dt));
```

everything is same except we are adding staging table.

```
sqoop export \
--connect jdbc:mysql://quickstart.cloudera:3306/banking \
--username root \
--password cloudera \
--table card_transactions \
--staging-table card_transactions_stage \
--export-dir /data/card_trans-200913-220429.csv \
--fields-terminated-by ","
```

SQOOP INCREMENTAL

1.APPEND MODE

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental append \
--check-column order_id \
--last-value 0
```

Incremental mode is append
Check column is order_id
Last value is 0

This will import all the data from 0 to 68883 rows.

Next time when you will import you will need to give these details as input,these details will be avilable in the logs when you import the data

```
--incremental append \
--check-column order_id \
--last-value 68883
```

Now lets say you have added 5 new records in orders table in MYSQL

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental append \
--check-column order_id \
--last-value 68883
```

Now this will add only new 5 records.

2. LAST MODIFIED MODE

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental lastmodified \
--check-column order_date \
--last-value 0 \
--append
```

When we use Last modified mode we need to give check column as date column.
This will import all the rows in first go.

Details to be filled in next import

```
--incremental lastmodified
--check-column order_date
--last-value 2022-09-22 03:20:09
```

```
select current_timestamp from dual;
```

Now add some records in the orders table and update and update any of the record.

```
insert into orders values(68890,current_timestamp,5523,'COMPLETE');
insert into orders values(68891,current_timestamp,5523,'COMPLETE');
insert into orders values(68892,current_timestamp,5523,'COMPLETE');
insert into orders values(68893,current_timestamp,5523,'COMPLETE');
insert into orders values(68894,current_timestamp,5523,'COMPLETE');
update orders set order_status='COMPLETE' and order_date = 2020-06-16 00:00:00 WHERE
ORDER_ID = 68862;
commit;
```

For next import-->

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental lastmodified \
--check-column order_date \
--last-value '2022-09-22 03:20:09' \
--append
```

if a record is updated in your table and then we use incremental import with last modified, then we will get updated record also

INCREMENTAL MERGE KEY

for e.g we updated record where order_id is 50000

Then in HDFS you will see
50000 oldtimestamp in hdfs
50000 newtimestamp in hdfs

you want hdfs file should be always in sync with the table(only updated record)

--append will result in duplicate records in HDFS

```
sqoop import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental lastmodified \
--check-column order_date \
--last-value '2022-09-22 03:20:09' \
--merge-key order_id
```

merge key will contain the column with primary key.
--merge key will only give the unique records with latest timestamp

SQOOP JOB

sqoop job will help you to remember the parameters.It will automate the things.

```
sqoop job \
--create job_orders \
-- import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--warehouse-dir /data \
--incremental append \
--check-column order_id \
--last-value 0
```

There is a space between -- & import.

To see the list of sqoop jobs.

```
sqoop job --list
```

Executing the sqoop job

```
sqoop job --exec job_orders
```

It will say--> Saving incremental import state to the metastore

For checking the saved state of job

```
sqoop job --show job_orders
```

You will get --> incremental.last.value = 68894

Deleting a sqoop job

```
sqoop job --delete job_orders
```

PASSWORD MANAGEMENT

How to create a password file?

```
echo -n "cloudera" >> .password-file
```

To see hidden file --> ls -a
To see content --> cat .password-file

FULLY AUTOMATIC JOB

```
sqoop job \
--create job_orders \
-- import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password-file file:///home/cloudera/.password-file  \
--table orders \
--warehouse-dir /data \
--incremental append \
--check-column order_id \
--last-value 0
```

file:// indicates that the password file is in local, and not in HDFS.
if you do not mention file:// then it will expect the file in HDFS.
Now during job execution it will not ask for password,it will be a fully automated process.

Where is state of the sqoop job stored?

locally in folder named (.sqoop)

ls -altr /home/cloudera
cd .sqoop ---> ls ---> metastore.db.script

```
cat metastore.db.script | grep incremental
```

Here you will see all the 3 parameters stored.

JCEKS (JAVA CRYPTOGRAPHY ENCRYPTION KEY STORE)

How to create a password alias

```
hadoop credential create mysql.banking.password -provider
jceks://hdfs/user/cloudera/mysql.password.jceks
```

It will ask to enter the password
shubham

mysql.banking.password = Name of password alias
jceks://hdfs/user/cloudera/mysql.password.jceks = Location where password is stored in encrypted form.

To check the content of file created

```
hadoop fs -cat /user/cloudera/mysql.password.jceks
```

It will be in encrypted form.

Try running this command if it works

```
sqoop eval \
-
Dhadoop.security.credential.provider.path=jceks://hdfs/user/cloudera/mysql.password.jcek
s \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password-alias mysql.banking.password \
--query "select count(*) from orders"
```

ADDITIONAL READING

CONCEPT 1 – DEFAULT IMPORT

By default, Sqoop will import a table named orders to a directory named orders inside your home directory in HDFS. For example, if your username is someuser, then the import tool will write to /user/someuser/orders/(files)

```
sqoop-import\
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db\
--username root\
--password cloudera NO
--table orders
```

Note: we are not specifying the target-dir or warehouse-dir.
Also just like we import a table we can import a view also.

CONCEPT 2 - FREE FORM QUERY IMPORT

Instead of using the --table, --columns and --where arguments, you can specify a SQL statement with the --query argument. Comment men

When importing a free-form query, you must specify a destination directory with --target-dir.

If you want to import the results of a query in parallel, then each map task will need to execute a copy of the query, with results partitioned by bounding conditions inferred by Sqoop. Your query must include the token $CONDITIONS which each Sqoop process will replace with a unique condition expression. You must also select a splitting column with --split-by.

Note: If you are issuing the query wrapped with double quotes ("), you will have to use \$CONDITIONS instead of just $CONDITIONS to disallow your shell from treating it as a shell variable.

Example 1:

```
sqoop-import\
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db\
--username root\
--password cloudera
--query 'select * from orders where $CONDITIONS AND order_id >50000' \
--target-dir /data/orders3\
--split-by order_id
```

Example 2:

```
sqoop-import\
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db\
--username root\
--password cloudera \
--query “select * from orders where \$CONDITIONS AND order_id >50000” \
--target-dir /data/orders3\
--split-by order_id
```

Note: The facility of using free-form query in the current version of Sqoop is limited to simple queries where there are no ambiguous projections and no OR conditions in the WHERE clause. Use of complex queries such as queries that have sub-queries or joins leading to ambiguous projections can lead to unexpected results.

CONCEPT 3 - DIRECT IMPORT

Controlling the Import Process

By default, the import process will use JDBC which provides a reasonable cross-vendor import channel. Some databases can perform imports in a more high-performance fashion by using database-specific data movement tools.

For example, MySQL provides the mysqldump tool which can export data from MySQL to other systems very quickly.

By supplying the --direct argument, you are specifying that Sqoop should attempt the direct import channel. T

his channel may be higher performance than using JDBC. But can be used for very basic things only.

Example:

```
sqoop-import \
--username root
--password cloudera \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--table orders \
--target-dir /data/orders \
--direct
```

CONCEPT 4 – VALIDATE

Validate the data copied, either import or export by comparing the row counts from the source and the target post copy.

Validation currently only validates data copied from a single table into HDFS and there are a lot of limitations.

Example:

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--target-dir /data/orders \
--validate
```

```
Data successfully validated
```

CONCEPT 5 – EXPORTS MAY FAIL FOR NUMBER OF REASONS.

1. Loss of connectivity from the Hadoop cluster to the database (either due to hardware fault, or server software crashes)

2. Attempting to INSERT a row which violates a consistency constraint (for example, inserting a duplicate primary key value)

3. Attempting to parse an incomplete or malformed record from the HDFS source data

4. Attempting to parse records using incorrect delimiters

5. Capacity issues (such as insufficient RAM or disk space)

Note: 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.

CONCEPT 6 – IMPORTING DATA INTO HBASE

Sqoop supports additional import targets beyond HDFS and Hive. Sqoop can also import records into a table in HBase.

By specifying --hbase-table, you instruct Sqoop to import to a table in in HBase rather than the directory in HDFS. Sqoop will import data to the table specified as the argument to –hbase-table.

Each row of the input table will be transformed into an HBase Put operation to a row of the output table. The key for each row is taken from a column of the input. By default, Sqoop will use the split-by column as the row key column. If that is not specified, it will try to identify the primary key column, if any, of the source table.

You can manually specify the row key column with --hbase-row-key. Each output column will be placed in the same column family, which must be specified with --column-family.

If the target table and column family do not exist, the Sqoop job will exit with an error. You should create the target table and column family before running an import. If you specify --hbase-create-table, Sqoop will create the target table and column family if they do not exist, using the default parameters from your HBase configuration.

CONCEPT 7 – IMPORTING DATA INTO HIVE

Sqoop's import tool's main function is to upload your data into files in HDFS. If you have a Hive metastore associated with your HDFS cluster, Sqoop can also import the data into Hive by generating and executing a CREATE TABLE statement to define the data's layout in Hive. Importing data into Hive is as simple as adding the --hive-import option to your Sqoop command line.

If the Hive table already exists, you can specify the --hive-overwrite option to indicate that existing table in hive must be replaced. After your data is imported into HDFS or this step is omitted, Sqoop will generate a Hive script containing a CREATE TABLE operation defining your columns using Hive's types, and a LOAD DATA INPATH statement to move the data files into Hive's warehouse directory.

Sqoop will by default import NULL values as string null. Hive is however using string \N to denote NULL values and therefore predicates dealing with NULL (like IS NULL) will not work correctly. You should append parameters --null-string and --null-non-string in case of import job or --input-null-string and --input-null-non-string in case of an export job if you wish to properly preserve NULL values. Because sqoop is using those parameters in generated code, you need to properly escape value \N to \\N:

```
sqoop import ... --null-string '\\N' --null-non-string '\\N'
```

The table name used in Hive is, by default, the same as that of the source table. You can control the output table name with the --hive-table option.

Example-

```
sqoop-import \
--connect jdbc:mysql://quickstart.cloudera:3306/retail_db \
--username root \
--password cloudera \
--table orders \
--hive-import \
--hive-table orders_new \
--verbose
```

FAQs

How is split size calculated when the primary key is not in sequence...I am referring to the example: sqoop import with where clause, where we only consider the records with order status as 'processing'. Hence order-id is not in sequence...is it not leading to unequal distribution of data among the mappers

It is always preferred to go with numeric columns that too where you have unique values. Split wise is always used in a column that has no outliers and is an indexed one.
Even when the primary key is not in sequence (when we apply where condition), mappers automatically divides the work equally among all 4 mappers.

What happens when the sqoop job fails while transferring a large transfer job?

If it is an export job with --staging-table, your staging table will hold partial data.

If it is an import job, you may find few mapper output files created in your target directory provided one or more mappers are complete. We will have to delete the directories which formed during import and restart the job.

What can be the Sqoop command if we wish to import only thefirst 10 records from a table?

You can use a limit clause in select statements while importing. Sqoop eval is preferred if you want to get the sample of data.

When we modify boundary query like "select 1, 68883" and wehave outliers in data (with index 200000) then we would end up pulling data without an outlier record right?

Correct. Outlier should be ingested using one more sqoop command

If our table has a COMPOSITE primary key then how sqoop IMPORT works?,I mean how boundary query will find min (primary key) and max (primary key)?

just like it does in case of string using hash value

Can we use --merge-key argument to merge all mapper output files?

It's not to merge files. rather to merge keys so that we do not have repeated keys. if you want to merge files then use getmerge

What is the meaning of Overriding argument of save job in sqoop and how it done by --exec

That means if you have to change any parameter value

How does split by works for non-numeric columns?

It will divide the data based on characters. But the issue comes when multiple entries start with the same characters encountered then it considers 2nd character and it goes on.

It becomes more complex when it finds uppercase, numbers, special characters etc. The Algorithm is very complex and sometimes unable to resolve conflicts resulting in repeating entries or missing entries. That's why it is not recommended.

how to use -e or -query argument with sqoop import? For example, if I want to import the data limit to 10 (e.g: select * from abc limit 10;) how these arguments can be used?

Your --query must include the token $CONDITIONS--query "select * from retail_db.orders where \$CONDITIONS LIMIT 10"

If the primary key is not number type, how will be the min and max for boundary val query be calculated?

Non numeric keys are not recommended as it will internally convert it to ASCII value for string n The even distribution of load among mappers is not guaranteed n job might fail as well.

How are sqoop queries scheduled in real PRODUCTION environments, Is it oozie or some other tool?

Oozie or Airflow

What if we want the outlier as well when importing? Suppose records are from 1 – 100 And 555 is outlier?

We are defining boundary query, not to process the outlier.
If you want all rows, don't define boundary query.

Distrubution among mappers will be like-
In that case split size = 139(138.5)
Mapper 1 will process order_id 1-139
Mapper 2 140 - 279
Mapper3 280 – 419
Mapper4 420 – 555

With above all records processes in mapper 1 (order_id)
mapper2 and mapper3 run with 0 records.
Mapper4 will run with one record order_id 555.

If the sqoop export fails and we are using the staging table, then the staging table will get partial data but the target table will not be impacted. Now what will happen if we run the sqoop export again, do we have to manually truncate the table or it will be automatically done?

--clear-staging-table will ensure that data is deleted in the staging table before the export

Is it feasible to use staging table while exporting large amount of data? performance prospective?

Not necessarily. We tried to export GBs of data, didn't find any issue. If you still have any performance issue, you increase the frequency of job. For example: rather than running every 4 hours, run it for every 2 hours.

Is there a way to import all columns of a table except one? I know we can use --columns and --query options. But is there something like --exclude-column?

I don't think there is a direct parameter in sqoop command to exclude columns. (If any please let me know). Only --exclude-tables is available to restrict certain tables while doing import-all-tables

In boundary val query if we are split by non-primary key column and it contains duplicates will all the records get imported?
For example - if there are two records invorder_items table for order_item_order_id = 68880

If it is not a Primary key and you have duplicates, I think that both rows with id 68880 should be imported to HDFS.
Please check the rows retrieved and count(*) from the table to make sure.
Cat all files with grep that 68880.
hadoop fs -cat/user/cloudera/bvqresult1/order_items/* | grep 68880

What is the efficient way to handle outlier data if that data is valid? I understand we can ignore it while customising boundary query, but in that case the record will be filtered out right. In the example given in video, the record with ID 200000 will not be transferred to HDFS.

First is let it run with default boundary val. but this will be inefficient from mappers point of view. Second approach is I can run 2 sqoop imports. Let's suppose you have date from 1 to 500 and then 10000 to 10003. First import I will process only 500 records and in second I can process the last 4. This way my imports will be faster and records will be distributed properly on mappers

When we import data from a mysql table to hdfs, how we will validate the table data properly loaded into the file or not?

We can use --validate in sqoop to validate the data. however, this will only check for the count of the rows from source and destination and not the data

Sqoop eval doesn't support --warehouse-dir argument

Yes, it doesn’t support warehouse-dir argument.

Sqoop supports data imported into following services:

- HDFS
- Hive
- Hbase
- Hcatalog
- Accumulo

Role of JDBC driver in sqoop setup? Is the JDBC driver enough to connect the sqoop to the database?

Sqoop needs a connector to connect the different relational databases. Almost all Database vendors make a JDBC connector available specific to that Database, Sqoop needs a JDBC driver of the database for interaction.
No, Sqoop needs JDBC and a connector to connect a database.

Define Sqoop metastore? What is the purpose of Sqoop-merge?

Sqoop meta store is a tool for using hosts in a shared metadata repository. Multiple users and remote users can define and execute saved jobs defined in metastore. End users configured to connect the metastore in sqoop-site.xml or with the
–meta-connect argument.

The purpose of sqoop-merge is:
This tool combines 2 datasets where entries in one dataset overwrite entries of an older dataset preserving only the new version of the records between both the data sets.

Explain the saved job process in Sqoop.

Sqoop allows us to define saved jobs which make this process simple. A saved job records the configuration information required to execute a Sqoop command at a later time. sqoop-job tool describes how to create and work with saved jobs. Job descriptions are saved to a private repository stored in $HOME/.sqoop/.

We can configure Sqoop to instead use a shared metastore, which makes saved jobs offered to multiple users across a shared cluster. Starting the metastore is covered by the section on the sqoop-metastore tool.

What are the basic commands in Hadoop Sqoop and its uses?

The basic commands of HadoopSqoop are

- Codegen, Create-hive-table, Eval, Export, Help, Import, Import-all-tables, List-databases, List-tables,Versions.
- Useof HadoopSqoop basic commands
- Codegen- It helps to generate code to interact with database records.
- Create-hive-table- It helps to Import a table definition into a hive
- Eval- It helps to evaluateSQL statement and display the results
- Export-It helps to export an HDFS directory into a database table
- Help- It helps to list the available commands
- Import- It helps to import a table from a database to HDFS
- Import-all-tables- It helps to import tables from a database to HDFS
- List-databases- It helps to list available databases on a server
- List-Tables-It helps to list tables in a database
- Version-It helps to display the version information

Is sqoop same as to distcp in hadoop?

No. Because the only distcp import command is same as Sqoop import command and both the commands submit parallel map-only jobs but both command functions are different. Distcp is used to copy any type of files from Local filesystem to HDFS and Sqoop is used for transferring the data records between RDBMS and Hadoop eco- system service.

Explain the significance of using –split-by clause in Apache Sqoop?

Split-by is a clause, it is used to specify the columns of the table which are helping to generate splits for data imports during importing the data into the Hadoop cluster. This clause specifies the columns and helps to improve the performance via greater parallelism. And also it helps to specify the column that has an even distribution of data to create splits, that data is imported.

How can you execute a free-form SQL query in Sqoop to import the rows in a sequential manner?

Ans. By using the –m 1 option in the Sqoop import command we can accomplish it. Basically, it will create only one **MapReduce** task which will then import rows serially

How will you list all the columns of a table using Apache Sqoop?

Features of Sqoop

Parallel import/export
While it comes to import and export the data, Sqoop uses YARN framework. Basically, that offers fault tolerance on top of parallelism.

Connectors for all major RDBMS Databases
However, for multiple RDBMS databases, Sqoop offers connectors, covering almost the entire circumference.

Import results of SQL query
Also, in **HDFS**, we can import the result returned from an SQL query.

Incremental Load
Moreover, we can load parts of table whenever it is updated. Since Sqoop offers the facility of the incremental load.

Full Load
It is one of the important features of sqoop, in which we can load the whole table by a single command in Sqoop. Also, by using a single command we can load all the tables from a database.

Kerberos Security Integration
Basically, Sqoop supports Kerberos authentication. Where Kerberos defined as a computer network authentication protocol. That works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

Load data directly into HIVE/HBase
Basically, for analysis, we can load data directly into Apache Hive. Also, can dump your data in **HBase**, which is a NoSQL database.

Compression
By using deflate(gzip) algorithm with –compress argument, We can compress your data. Moreover, it is also possible by specifying –compression-codec argument. In addition, we can also load compressed table in **Apache Hive.**

Support for Accumulo
It is possible that rather than a directory in HDFS we can instruct Sqoop to import the table in Accumulo.

What is the advantage of using –password-file rather than -P option while preventing the display of password in the sqoop import statement?

Inside a sqoop script, we can use The –password-file option. Whereas the -P option reads from standard input, preventing automation.

Is JDBC driver enough to connect sqoop to the databases?

No. to connect to a database Sqoop needs both JDBC and connector.

Use of Codegen command in Hadoop sqoop?

Basically, Codegen command generates code to interact with database records

What is a disadvantage of using –direct parameter for faster data load by sqoop?

The native utilities used by databases to support faster load do not work for binary data formats like Sequence File.

How will you update the rows that are already exported?

Basically, to update existing rows we can use the parameter –update-key. Moreover, in it, a comma-separated list of columns is used which uniquely identifies a row. All of these columns are used in the WHERE clause of the generated UPDATE query. All other table columns will be used in the SET part of the query.

What is Sqoop Validation?

It means to validate the data copied. Either import or export by comparing the row counts from the source as well as the target post copy. Likewise, we use this option to compare the row counts between source as well as the target just after data imported into HDFS. Moreover, While during the imports, all the rows are deleted or added, Sqoop tracks this change. Also updates the log file.

What is Purpose to Validate in Sqoop?

In Sqoop to validate the data copied is Validation main purpose. Basically, either Sqoop import or Export by comparing the row counts from the source as well as the target post copy.