Module 7

Understanding Sqoop

Thanachart Numnonda, Executive Director, IMC Institute

Thanisa Numnonda, Faculty of Information Technology,

King Mongkut's Institute of Technology Ladkrabang

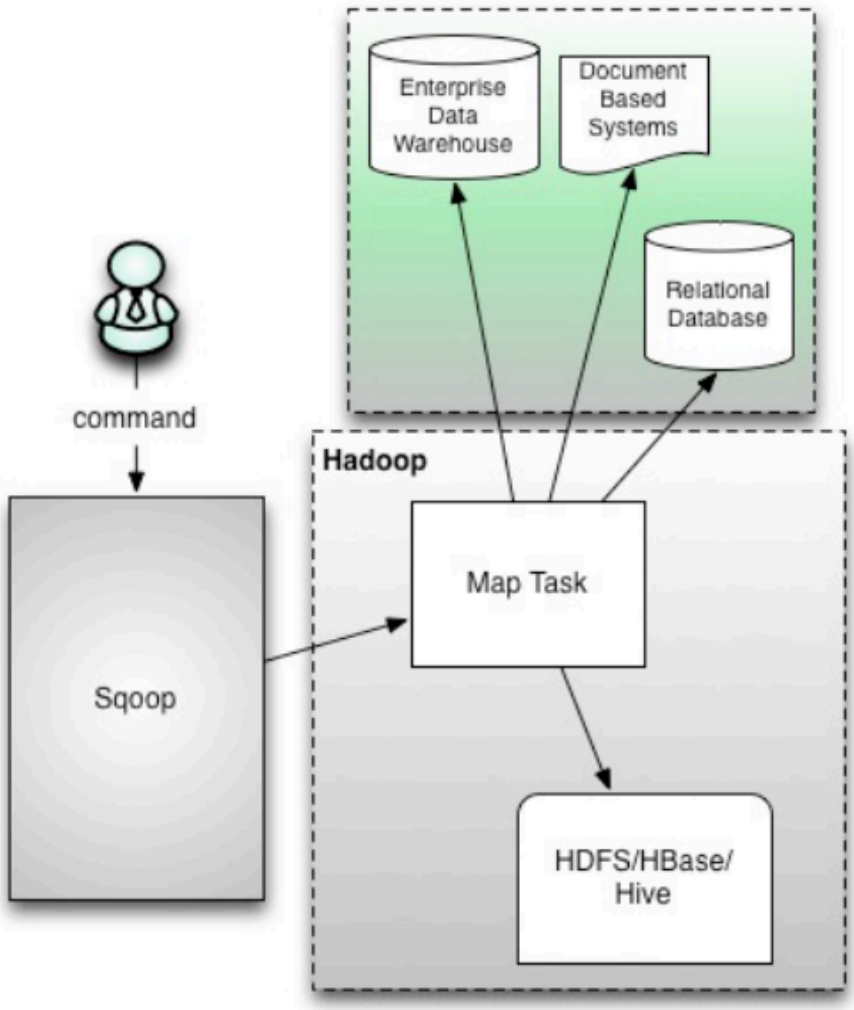
Introduction

Sqoop ("SQL-to-Hadoop") is a straightforward command-line tool with the following capabilities:

- **Imports individual tables or entire databases to files in HDFS**
- **Generates Java classes to allow you to interact with your imported data**
- **Provides the ability to import from SQL databases straight into your Hive data warehouse**

See also: http://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html

Architecture Overview



Sqoop Benefit

- Leverages RDBMS metadata to get the column data types
- It is simple to script and uses SQL
- It can be used to handle change data capture by importing
  daily transactional data to Hadoop
- It uses MapReduce for export and import that enables parallel and efficient data movement

Sqoop Mode

- Sqoop import: Data moves from RDBMS to Hadoop
- Sqoop export: Data moves from Hadoop to RDBMS

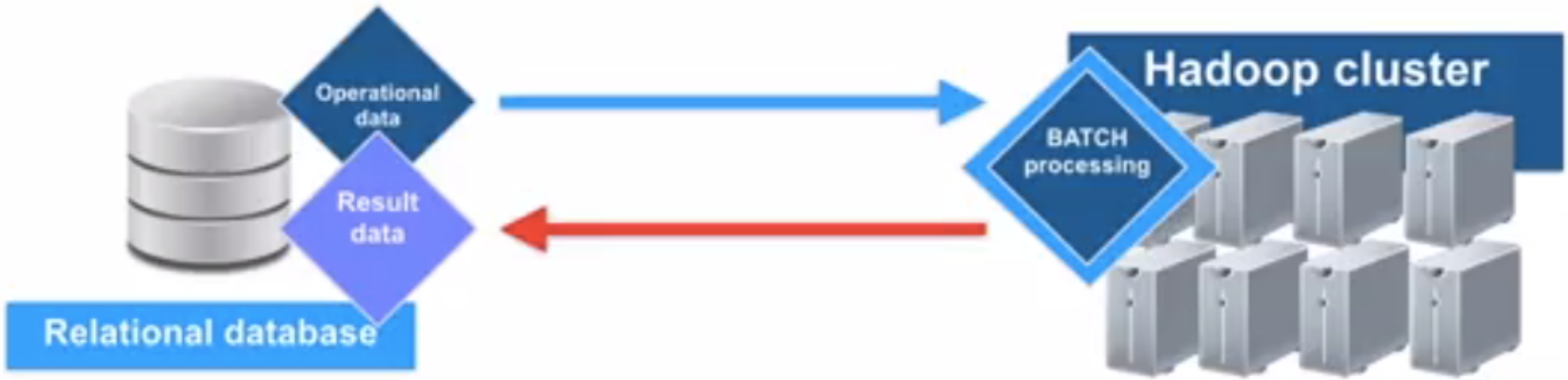
Use Case #1: ETL for Data Warehouse

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

Source: Mastering Apache Sqoop, David Yahalom, 2016

Use Case #2: ELT

- Extract operational data from RDBMS, process in Hadoop, return ***result*** to RDBMS



Use Case #3: Data Analysis

- Copy real-time data from RDBMS, combine with raw data on Hadoop using complex data analysis logic (not just SQL!)

Source: Mastering Apache Sqoop, David Yahalom, 2016

Use Case #4: Data Archival

- Move data from RDBMS after it expires to Hadoop, keeping the RDBMS "clean and lean"

Source: Mastering Apache Sqoop, David Yahalom, 2016

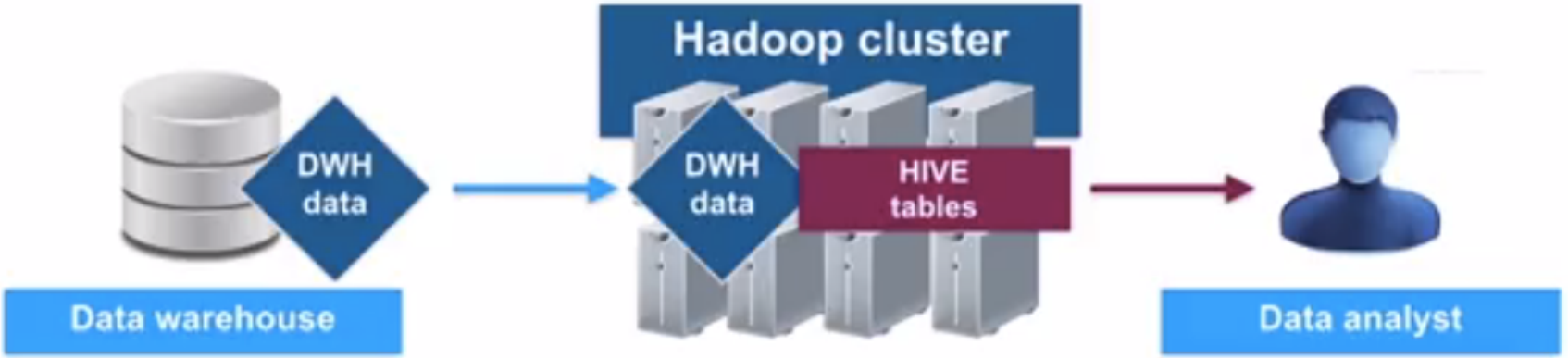
Use Case #5: Data Consolidation

- Integrate data from various organizational "data stores" to Hadoop for various data processing requirements

Source: Mastering Apache Sqoop, David Yahalom, 2016

Use Case #6: Move reports to Hadoop

- Easily allow traditional data analysis and business intelligence using Hadoop's power



Source: Mastering Apache Sqoop, David Yahalom, 2016

Import Commands

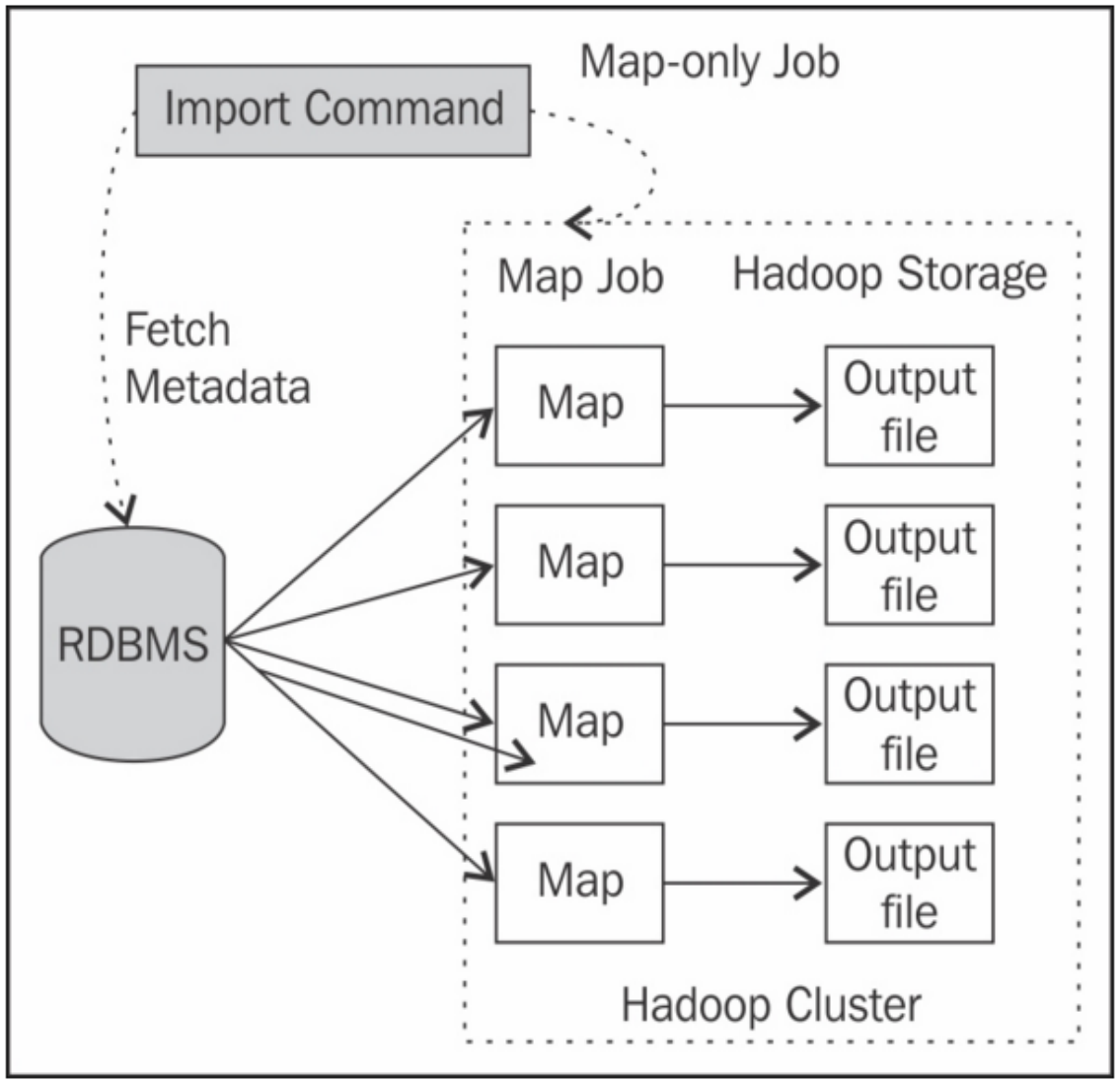
`--connect <jdbc-uri>`	Specifies the server or database to connect to. It also specifies the port. For example: `--connect jdbc:mysql://host:port/databaseName`
`--connection-manager <class-name>`	Specifies the connection manager class name.
`--driver <class-name>`	Specifies the fully qualified name of the JDBC driver class.
`--hadoop-home <dir>`	This parameter is used to override the `$HADOOP_HOME` environment variable.

Import Commands

`--password <password>`	Sets the authentication password required to connect to the input source.
`--username <username>`	Sets the authentication username.
`--connection-param-file <properties-file>`	Specifies the connection parameter's file.
`--help`	This option will provide the usage instructions.
`--verbose`	Prints more information during a query execution.

If a user doesn't want to specify the database password along with the command, the -P option can be used to read the password from the console.

Architecture of the import process



Incremental import

Parameter/argument	Description
`--check-column <column-name>`	The value of this column is used to determine the rows to be imported during the import process.
`--incremental <incremental-type>`	Specifies the type of incremental mode. Possible values are `append` and `lastmodified`.
`--last-value <value>`	Specifies the last value or the maximum value of the `check` column from the previous import. All the records whose `check` column value is greater than the value of the `–last-value` argument will be imported to HDFS.

```
bin/sqoop import -connect jdbc:mysql://localhost:3306/db1 -username root -password password -
table student -target-dir /user/abc/student -columns "student_id,address,name"  --incremental
lastmodified -last-value "2012-11-06 19:01:35"-check-column col4
```

Export Commands

Parameters	Description
`--direct`	Use the direct mode to perform the export quickly. Note that it is only supported for MySQL.
`--export-dir<dir>`	The location of input files in HDFS.
`--table <table-name>`	Name of the output table (the RDBMS table).
`-m,--num-mappers <n>`	Refers to the number of map tasks.

Export Commands

`--update-mode <mode>`	Specifies how updates are performed when new rows are found with non-matching keys in the database. Legal values for the mode include `updateonly` (default) and `allowinsert`.
`--update-key <col-name>`	The value of this column is used to identify the records that a user wants to update during the update mode. Use a comma-separated list of columns if there is more than one column.
`--staging-table <staging-table-name>`	Specifies the name of the staging table. The staging table is used to stage the data before inserting it into the destination table.
`--clear-staging-table`	This argument is used to clean the data from the staging table.

Hands-On: Loading Data from RDBMS to Hadoop

Running MySQL Docker

- **UbuntuCmd:** sudo docker pull mysql
- **UbuntuCmd:** sudo docker run --name mysqlServer -e MYSQL_ROOT_PASSWORD=itkmitl -p 3306:3306 -d mysql
- **UbuntuCmd:** sudo docker exec -it mysqlServer bash

```
root@f1922a70e09c:/# mysql -uroot -p"itkmitl"
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Prepare a test database table

```
mysql> CREATE DATABASE itkmitl_db;
mysql> USE itkmitl_db;
mysql> CREATE TABLE country_tbl(id INT NOT NULL, country
VARCHAR(50), PRIMARY KEY  (id));

mysql> INSERT INTO country_tbl VALUES(1, 'USA');
mysql> INSERT INTO country_tbl VALUES(2, 'CANADA');
mysql> INSERT INTO country_tbl VALUES(3, 'Mexico');
mysql> INSERT INTO country_tbl VALUES(4, 'Brazil');
mysql> INSERT INTO country_tbl VALUES(61, 'Japan');
mysql> INSERT INTO country_tbl VALUES(65, 'Singapore');
mysql> INSERT INTO country_tbl VALUES(66, 'Thailand');
```

View data in the table

```
mysql> SELECT * FROM country_tbl;
```

```
+----+-----------+
| id | country   |
+----+-----------+
|  1 | USA       |
|  2 | CANADA    |
|  3 | Mexico    |
|  4 | Brazil    |
| 61 | Japan     |
| 65 | Singapore |
| 66 | Thailand  |
+----+-----------+
7 rows in set (0.00 sec)
```

```
mysql> exit;
```

Then exit from the container by press Ctrl-P & Ctrl-Q

Inspect imcMysql IP Address

UbuntuCmd: sudo docker inspect mysqlServer | grep IPAddress

```
"IPAddress": "172.17.0.7",
```

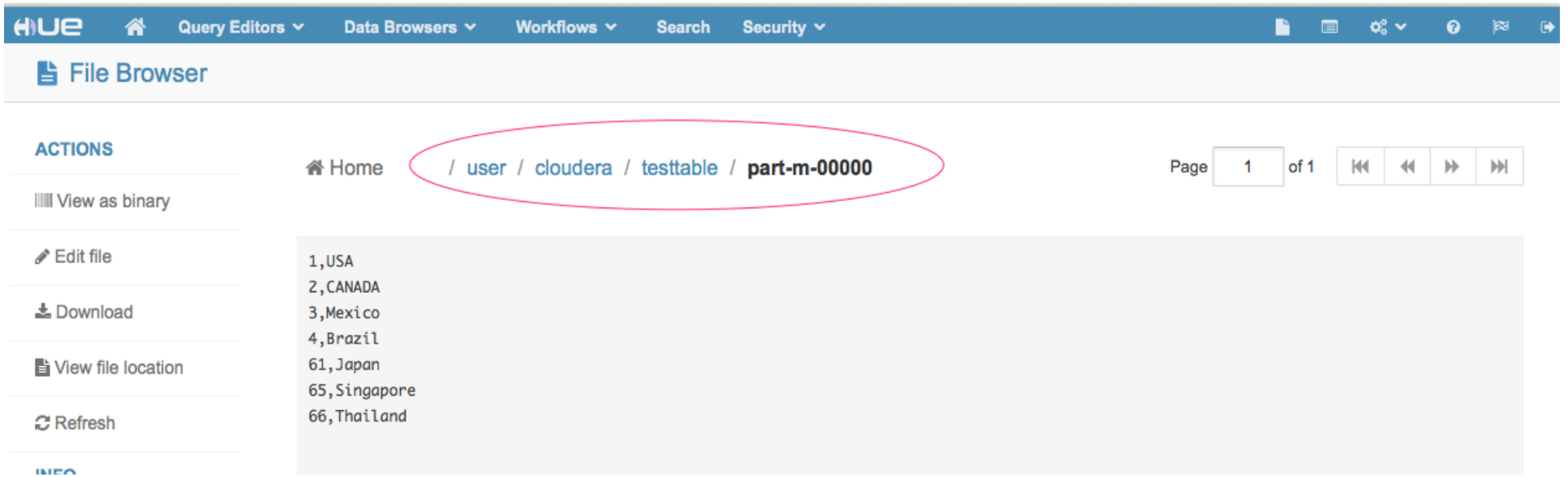
Restart the Cloudera docker with linking to the MySQL Docker

UbuntuCmd: `sudo docker run --hostname=quickstart.cloudera --privileged=true --link mysqlServer:mysqldb -t -i -p 8888:8888 cloudera/quickstart /usr/bin/docker-quickstart`

- *If both of these Dockers are up and running, you can find out the internal IP address of each of them by running this command. This gets the IP for mysqlServer.*

Case I: Importing data from MySQL to HDFS

```
[root@quickstart /]# sqoop import -connect jdbc:mysql://172.17.0.7/
itkmitl_db --username root --password itkmitl --table
country_tbl --target-dir /user/cloudera/testtable -m 1
```



Case II: Importing data from MySQL to HBase

```
$sqoop import --connect jdbc:mysql://172.17.0.7/itkmitl_db --
username root --password itkmitl --table country_tbl --hbase-
table country --column-family hbase_country_cf --hbase-row-key id --
hbase-create-table  -m 1
```

Start HBase

```
$hbase shell
hbase(main):001:0> list
```

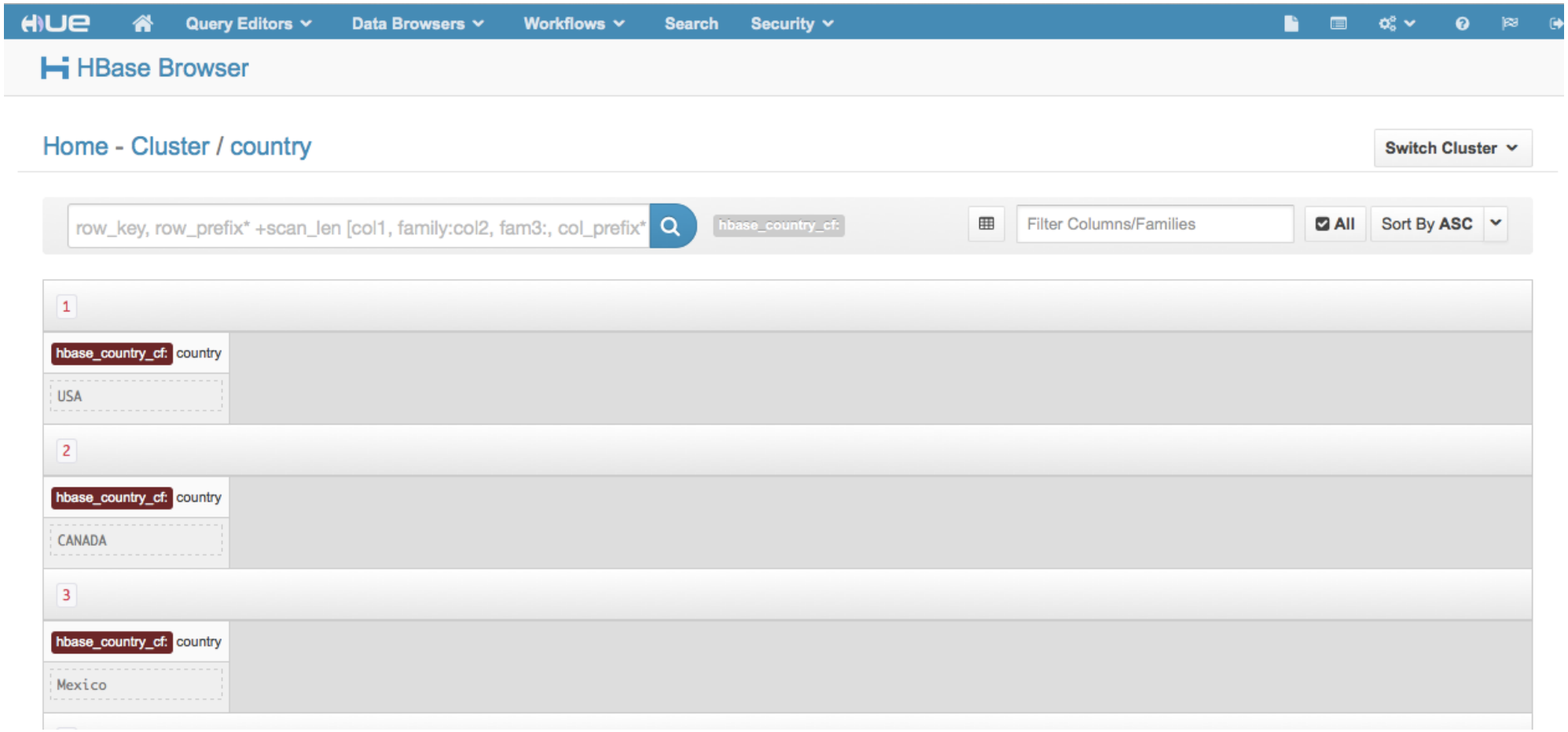
```
TABLE
country
1 row(s) in 0.2570 seconds

=> ["country"]
```

Viewing Hbase data

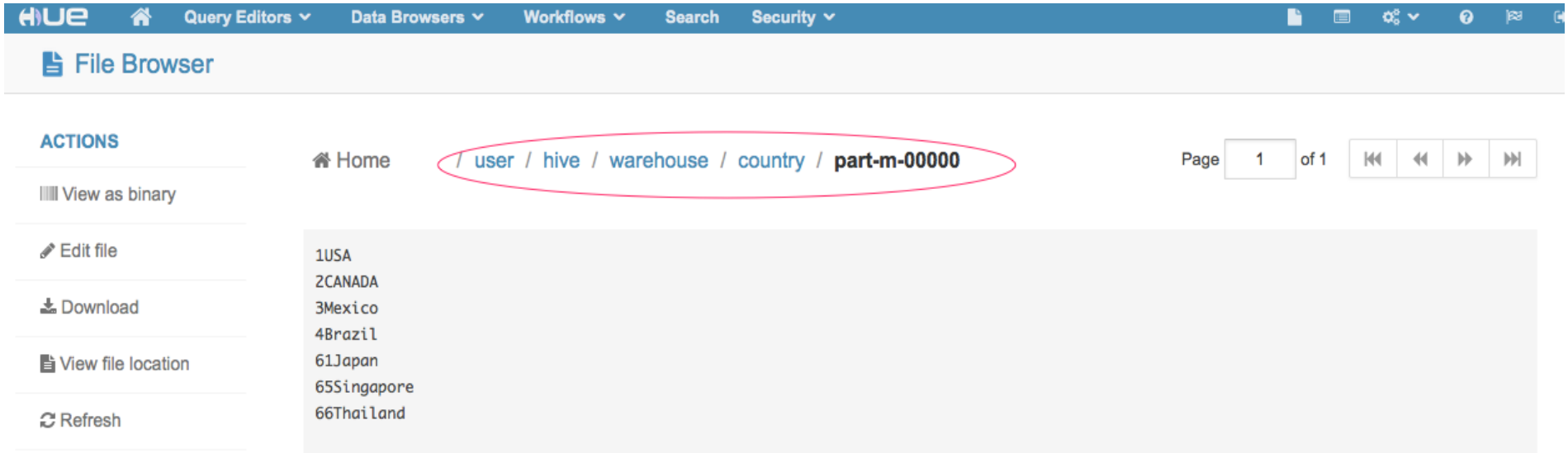
```
hbase(main):003:0> scan 'country'
ROW                     COLUMN+CELL
 1                      column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=USA
 2                      column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=CANADA
 3                      column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=Mexico
 4                      column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=Brazil
 61                     column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=Japan
 65                     column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=Singapore
 66                     column=hbase_country_cf:country, timestamp=1468081466623, val
                        ue=Thailand
7 row(s) in 0.1670 seconds
```

Viewing data from Hbase browser



Case III: Importing data from MySQL to Hive Table

```
$sqoop import --connect jdbc:mysql://172.17.0.7/itkmitl_db --
username root --password itkmitl --table country_tbl --hive-
import --hive-table country -m 1
```



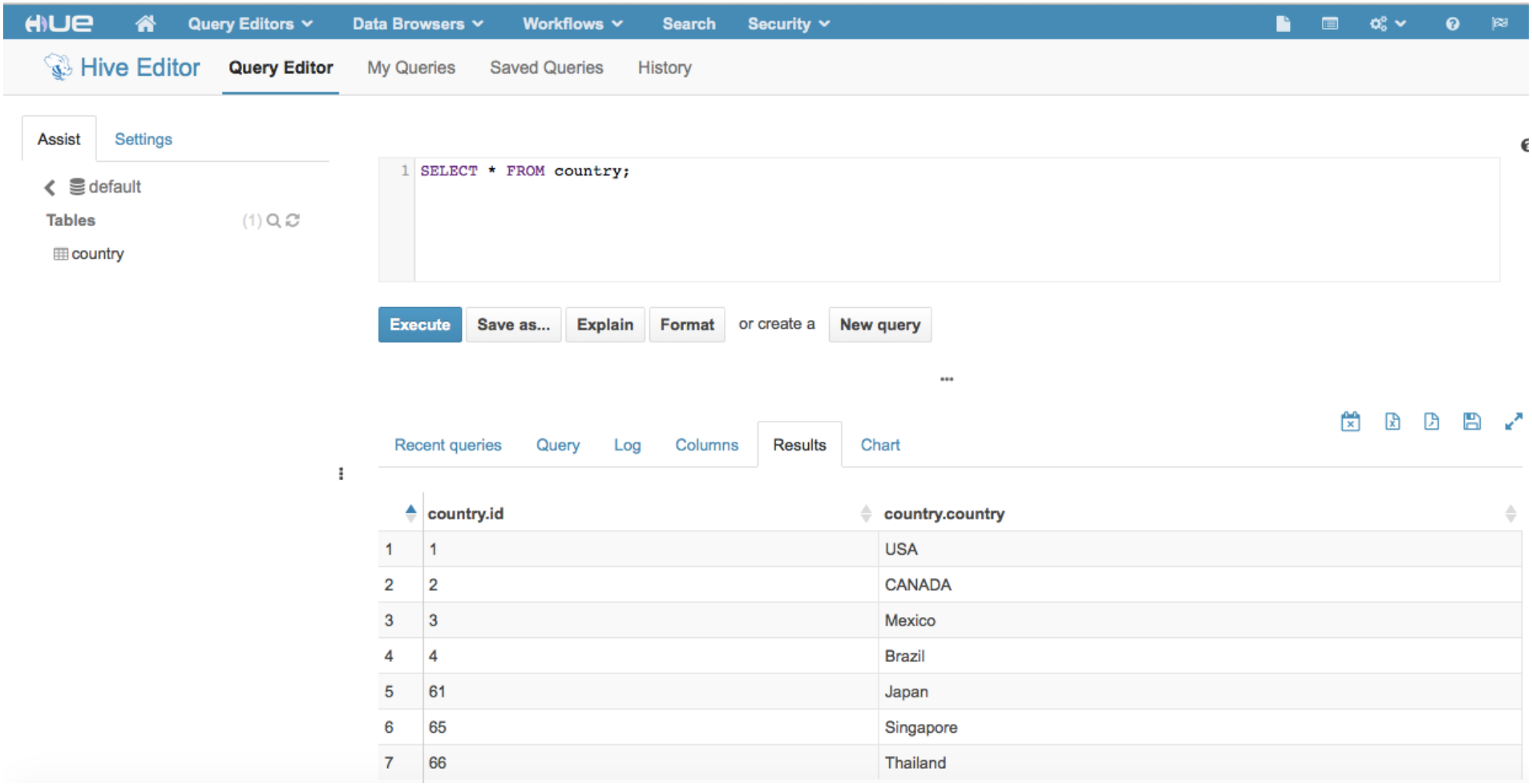
Reviewing data from Hive Table

```
[root@quickstart /]# hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.prope
rties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.

hive> show tables;
hive> select * from country;
```

```
1       USA
2       CANADA
3       Mexico
4       Brazil
61      Japan
65      Singapore
66      Thailand
Time taken: 0.587 seconds, Fetched: 7 row(s)
```

Running from Hue: Beewax



	country.id	country.country
1	1	USA
2	2	CANADA
3	3	Mexico
4	4	Brazil
5	61	Japan
6	65	Singapore
7	66	Thailand

Hands-On: Exporting Data from Hadoop to RDBMS

Exporting data from HDFS to MySQL

```
$wget https://s3.amazonaws.com/imcbucket/data/country.txt
$hdfs dfs -put country.txt /user/cloudera/input/
$sqoop export --connect jdbc:mysql://172.17.0.7/itkmitl_db --
username root --password itkmitl --table country_tbl --
export-dir /user/cloudera/input/country.txt
```

Viewing Result in a MySQL table on a Ubuntu server

```
UbuntuCmd: sudo docker exec -it mysqlServer bash
root@f1922a70e09c:/# mysql -uroot -p"itkmitl"
mysql> USE itkmitl_db;
mysql> SELECT * FROM country_tbl;
```