Loading Incremental Data into HDFS using Sqoop

Use case

Version 2.0

Loading Incremental Data into HDFS using Sqoop

Table of Contents

Loading Incremental Data into HDFS using Sqoop

Problem Statement:

We know how to transfer the content of entire table of a relational database into HDFS using Sqoop.

Let us assume that we have new records/data getting added into the table of a relational database day by day. When we reimport the data into HDFS, content of entire table will be loaded into HDFS which includes both old and new data which is not an optimal solution. The amount of time needed to import the data will increase in proportion with the data added newly every day which will degrade the performance as well. To overcome this, Sqoop has introduced incremental imports which appends the new data to the old data present in HDFS (i.e. it won't reimport the old data).

Let us see how to import the incremental data into HDFS.

Important Links:

Edureka VM Installation:
Please refer to Installation guide section present in the LMS for accessing the Edureka VM Installation Guide.

Codes along with the Dataset:

Dataset:

Let us consider the below sample two records as two datasets.

Initial Data in the table:

```
1, "Freemont", "1983-05-22 01:01:01"
```

New Data added in the table:

```
2, "Jicin", "1987-02-02 02:02:02"
```

Dataset Description:

The above data indicates the user information who visits a website. It consists of three fields.

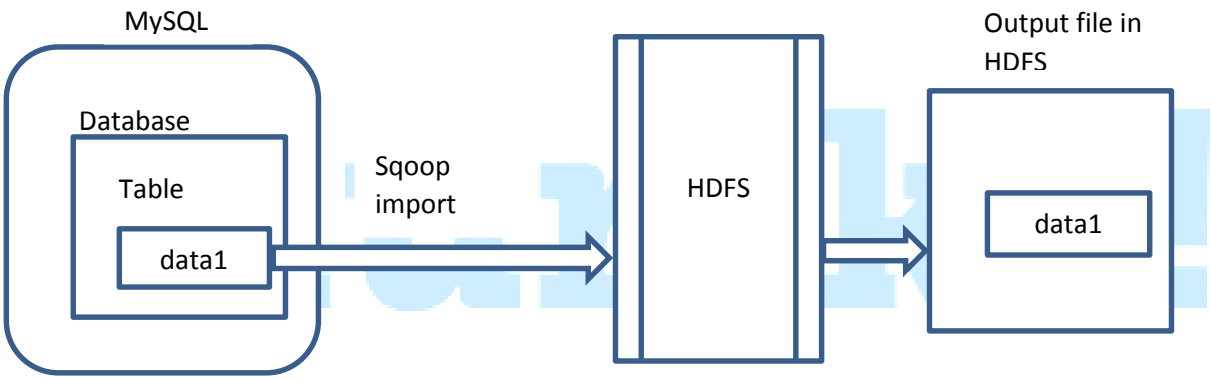
- Id
- Name
- Last update date-time

Tools and Technologies used:

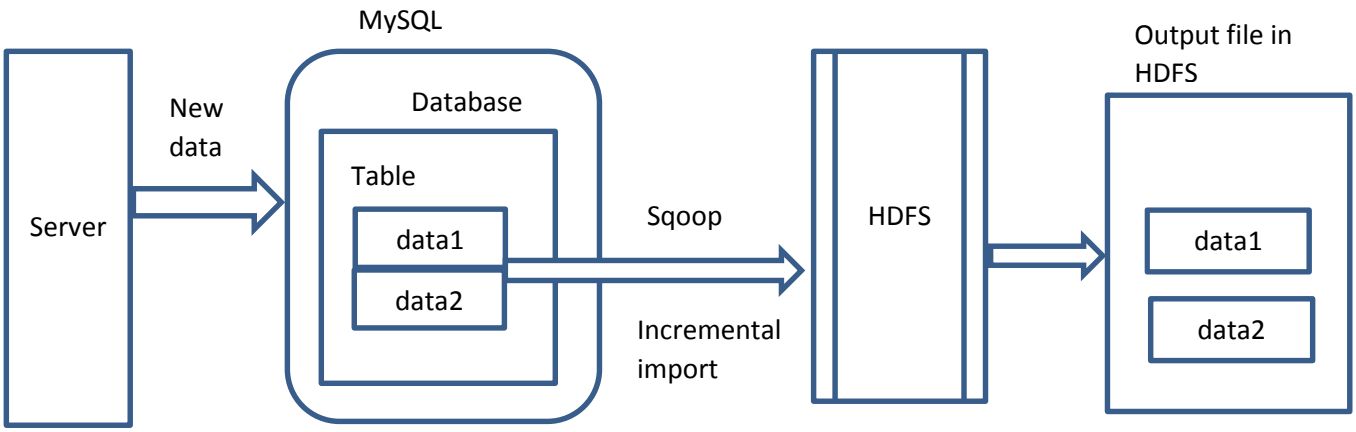
- Sqoop
- MySQL

Dataflow Diagram:

Trying to import the data into HDFS:



Trying to import incremented data into HDFS:



Implementation:

First let us create a table in MySQL, load some data and import that data to HDFS using Sqoop.

Command: mysql -u root

```
[edureka@localhost ~]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

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
```


Command: create database db;

Command: use db;

Command: CREATE TABLE visits (

id INTEGER UNSIGNED NOT NULL,

city VARCHAR(50),

last_update_date DATETIME NOT NULL,

PRIMARY KEY (id),

KEY (last_update_date)

);

Command:

INSERT INTO visits(id, city, last_update_date) VALUES(1, "Freemont", "1983-05-22 01:01:01");

We should have a primary key in the table to import incremental data which helps us to sync the data that exists in HDFS

```
mysql> create database db;
Query OK, 1 row affected (0.01 sec)

mysql> use db;
Database changed
mysql> CREATE TABLE visits (
    ->   id INTEGER UNSIGNED NOT NULL,
    ->   city VARCHAR(50),
    ->   last_update_date DATETIME NOT NULL,
    ->   PRIMARY KEY (id),
    ->   KEY (last_update_date)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO visits(id, city, last_update_date)
    -> VALUES(1, "Freemont", "1983-05-22 01:01:01");
Query OK, 1 row affected (0.00 sec)
```

Note: We should have a database table with a primary key to append new rows and to periodically sync the table's state to Hadoop for further processing. In this example, we have considered id as INTEGER Primary Key.

Command: select * from visits;

Checking if the data loaded into table.

```
mysql> select * from visits;
+----+----------+---------------------+
| id | city     | last_update_date    |
+----+----------+---------------------+
|  1 | Freemont | 1983-05-22 01:01:01 |
+----+----------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Command: grant all privileges on *.* to root@localhost identified by 'edureka' with grant option;

Granting privileges to root user

```
mysql> grant all privileges on *.* to root@localhost identified by 'edureka' with grant o
ption;
Query OK, 0 rows affected (0.00 sec)
```

Let us import this data into HDFS using sqoop.

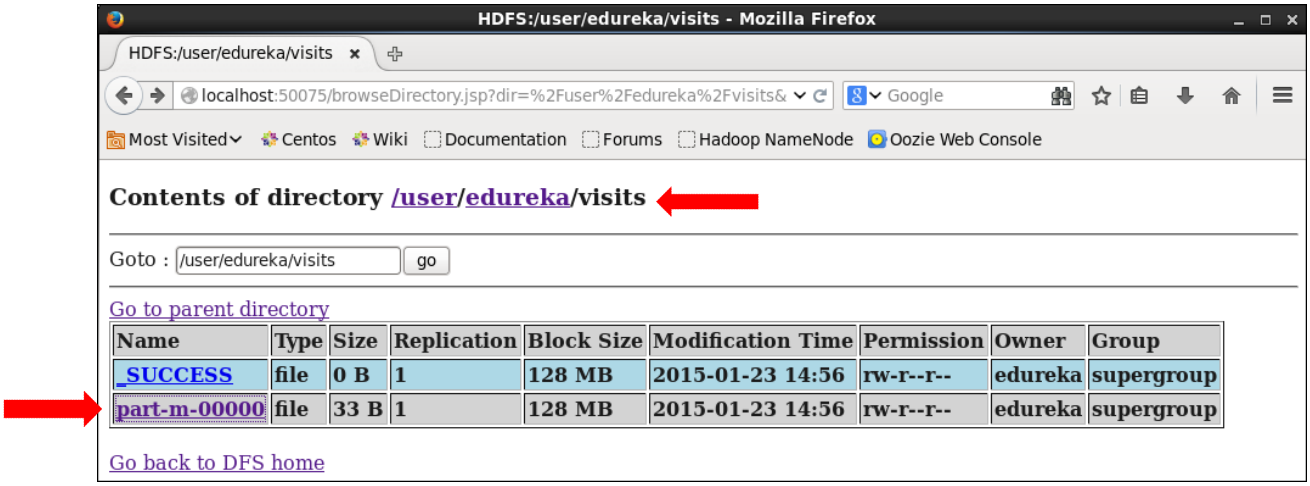
Command: sqoop import --connect jdbc:mysql://localhost/db --username root --password edureka --table visits

```
[edureka@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/db --username root -
-password edureka --table visits
Warning: /usr/lib/hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
15/01/23 14:30:03 WARN tool.BaseSqoopTool: Setting your password on the command-line is i
nsecure. Consider using -P instead.
15/01/23 14:30:03 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset
.
15/01/23 14:30:03 INFO tool.CodeGenTool: Beginning code generation
15/01/23 14:30:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `visi
ts` AS t LIMIT 1
15/01/23 14:30:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `visi
ts` AS t LIMIT 1
15/01/23 14:30:04 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-2.2.
0
Note: /tmp/sqoop-edureka/compile/1ddae736f33f1334dadd716098c4c33c/visits.java uses or ove
rrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
15/01/23 14:30:05 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-edureka/compi
le/1ddae736f33f1334dadd716098c4c33c/visits.jar
```

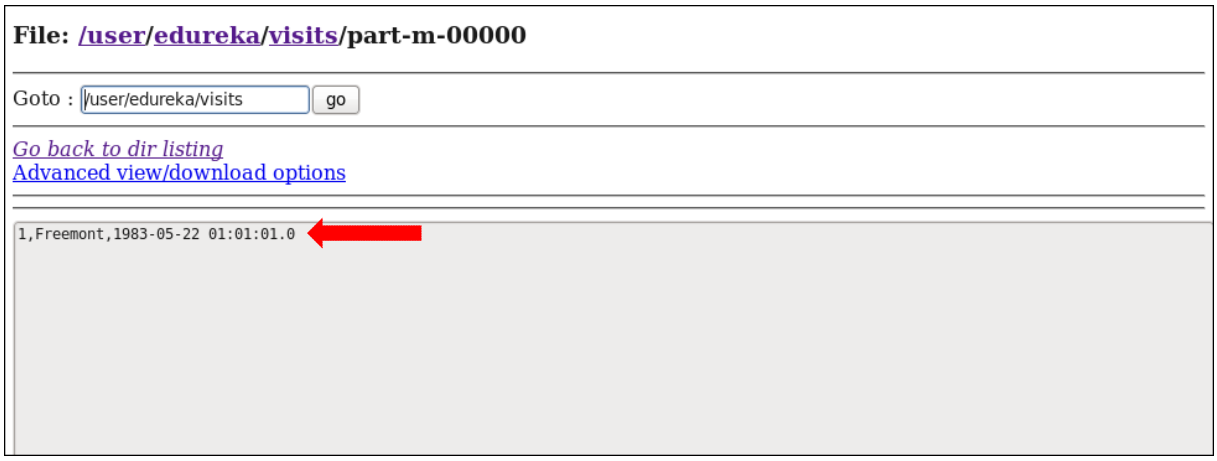
```
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2459
                Total time spent by all reduces in occupied slots (ms)=0
        Map-Reduce Framework
                Map input records=1
                Map output records=1
                Input split bytes=99
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=57
                CPU time spent (ms)=530
                Physical memory (bytes) snapshot=65126400
                Virtual memory (bytes) snapshot=368701440
                Total committed heap usage (bytes)=16252928
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=33
15/01/23 14:30:18 INFO mapreduce.ImportJobBase: Transferred 33 bytes in 11.8595 seconds (
2.7826 bytes/sec)
15/01/23 14:30:18 INFO mapreduce.ImportJobBase: Retrieved 1 records.
[edureka@localhost ~]$
```

Data has been imported

Let us cross check if the data is imported correctly or not. As we have not mentioned target-dir in the above sqoop command it by defaults store in the directory /user/edureka with table name as filename.



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 B	1	128 MB	2015-01-23 14:56	rw-r--r--	edureka	supergroup
part-m-00000	file	33 B	1	128 MB	2015-01-23 14:56	rw-r--r--	edureka	supergroup



Now let us load one more record into MySQL table which acts as new data

Command: INSERT INTO visits(id, city, last_update_date) VALUES(2, "Jicin", "1987-02-02 02:02:02");

```
mysql> INSERT INTO visits(id, city, last_update_date)
    -> VALUES(2, "Jicin", "1987-02-02 02:02:02");
Query OK, 1 row affected (0.00 sec)

mysql>
```

Inserting the new data into table

Command: select * from visits;

Now the table has old data + new data

```
mysql> select * from visits;
+----+----------+---------------------+
| id | city     | last_update_date    |
+----+----------+---------------------+
|  1 | Freemont | 1983-05-22 01:01:01 |
|  2 | Jicin    | 1987-02-02 02:02:02 |
+----+----------+---------------------+
2 rows in set (0.00 sec)

mysql>
```

Let us perform the sqoop incremental import to append the newly added data to HDFS. To do that we need three more parameters

--incremental — as we want to import only new rows without changing the existing ones, we need to use the ***append*** mode.

--check-column — a column name that should be checked for newly appended data. Usually it would be the column which acts as primary key in the table. In this case, it would be ***id***

--last-value — contains the last value that successfully imported into Hadoop. In this case the last value of column id imported to HDFS was ***1***

Command: sqoop import --connect jdbc:mysql://localhost/db --username root --password edureka --table visits --incremental append --check-column id --last-value 1

Sqoop Incremental import

```
[edureka@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/db --username root -
-password edureka --table visits --incremental append --check-column id --last-value 1
Warning: /usr/lib/hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
15/01/23 15:00:44 WARN tool.BaseSqoopTool: Setting your password on the command-line is i
nsecure. Consider using -P instead.
15/01/23 15:00:44 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset
.
15/01/23 15:00:44 INFO tool.CodeGenTool: Beginning code generation
15/01/23 15:00:44 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `visi
ts` AS t LIMIT 1
15/01/23 15:00:44 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `visi
ts` AS t LIMIT 1
15/01/23 15:00:44 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/hadoop-2.2.
0
Note: /tmp/sqoop-edureka/compile/4ceb9be3d63cd7575c89727957ecd0ef/visits.java uses or ove
rrides a deprecated API.
```

```
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=30
15/01/23 15:00:57 INFO mapreduce.ImportJobBase: Transferred 30
2.8419 bytes/sec)
15/01/23 15:00:57 INFO mapreduce.ImportJobBase: Retrieved 1
15/01/23 15:00:57 INFO util.AppendUtils: Appending to direct
15/01/23 15:00:57 INFO util.AppendUtils: Using found partitio
15/01/23 15:00:57 INFO tool.ImportTool: Incremental import complet                incre
mental import of all data following this import, supply the following                uments:
15/01/23 15:00:57 INFO tool.ImportTool:  --incremental append
15/01/23 15:00:57 INFO tool.ImportTool:   --check-column id
15/01/23 15:00:57 INFO tool.ImportTool:   --last-value 2
15/01/23 15:00:57 INFO tool.ImportTool: (Consider saving this with 'sqoop job --create')
[edureka@localhost ~]$
```

Sqoop job ran succesfully

When you want to do incremental import again you can use these parameters.

Below is the structure of file stored in HDFS after performing Incremental import.

Part-m-00000 will consist of old data and Part-m-00001 will consist of new data

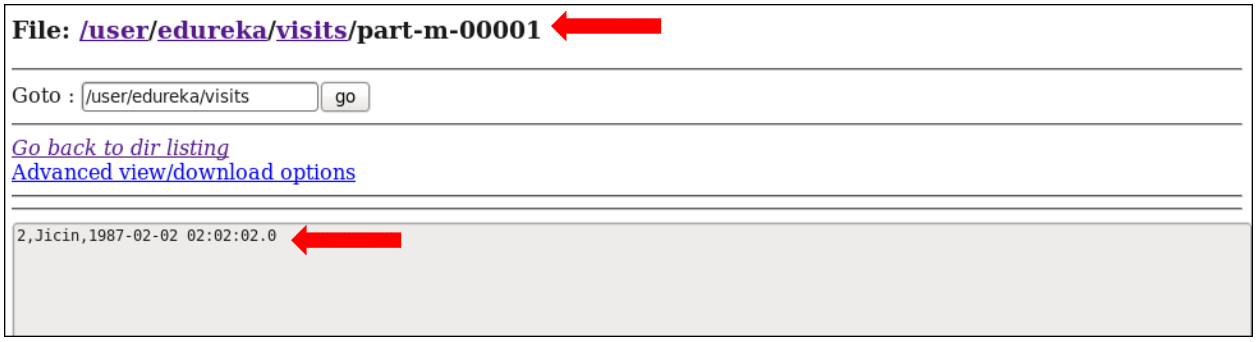
Contents of directory /user/edureka/visits

Goto : /user/edureka/visits go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 B	1	128 MB	2015-01-23 14:56	rw-r--r--	edureka	supergroup
part-m-00000	file	33 B	1	128 MB	2015-01-23 14:56	rw-r--r--	edureka	supergroup
part-m-00001	file	30 B	1	128 MB	2015-01-23 15:00	rw-r--r--	edureka	supergroup

Let us check the appended data.



We have successfully loaded incremental data into HDFS using Sqoop!!!

References:

Apache Sqoop Cookbook by Kathleen Ting & Jarek Jarcec Cecho

http://sqoop.apache.org/docs/1.4.3/SqoopUserGuide.html#_incremental_imports