**Schema evolution with Hive and Avro (Hive 0.14 and later versions)**

In production, we have to change the table structure to address new business requirements. The table schema has to change to add/delete/rename table columns. Any of these changes affect downstream ETL jobs adversely. In order avoid these, we have to make corresponding changes to ETL jobs and target tables.

Schema evolution allows you to update the schema used to write new data while maintaining backwards compatibility with the schemas of your old data. Then you can read it all together as if all of the data has one schema. Please read more on Avro serialization at the following URL: https:/ / avro. apache. org/ .

In the following example, we will demonstrate how Avro and Hive tables absorb the changes of source table's schema changes without ETL job failure.

We will create a customer table in the MySQL database and load it to the target Hive external table using Avro files. Then we will add one more column to the source tables to see how a Hive table absorbs that change without any errors.

Connect to MySQL to create a source table (customer):

mysql -u root -p

GRANT ALL PRIVILEGES ON \*.\* TO 'sales'@'localhost' IDENTIFIED BY 'xxx';

mysql -u sales -p

mysql> create database orders;

mysql> use orders;

CREATE TABLE customer(

cust\_id INT ,

cust\_name VARCHAR(20) NOT NULL,

cust\_city VARCHAR(20) NOT NULL,

PRIMARY KEY ( cust\_id )

);

Insert records into the customer table:

INSERT into customer (cust\_id,cust\_name,cust\_city) values (1,'Sam James','Austin');

INSERT into customer (cust\_id,cust\_name,cust\_city) values (2,'Peter Carter','Denver');

INSERT into customer (cust\_id,cust\_name,cust\_city) values (3,'Doug Smith','Sunnyvale');

INSERT into customer (cust\_id,cust\_name,cust\_city) values (4,'Harry Warner','Palo Alto');

On Hadoop, run the following sqoop command to import the customer table and store data in Avro files into HDFS:

hadoop fs -rmr /user/sqoop\_data/avro

sqoop import -Dmapreduce.job.user.classpath.first=true

--connect jdbc:mysql://localhost:3306/orders

--driver com.mysql.jdbc.Driver

--username sales --password xxx

--target-dir /user/sqoop\_data/avro

--table customer

--as-avrodatafile

Verify the target HDFS folder:

[root@sandbox ~]# hadoop fs -ls /user/sqoop\_data/avro

Found 7 items

-rw-r--r-- 1 root hdfs 0 2017-09-09 08:57 /user/sqoop\_data/avro/\_SUCCESS

-rw-r--r-- 1 root hdfs 472 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00000.avro

-rw-r--r-- 1 root hdfs 475 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00001.avro

-rw-r--r-- 1 root hdfs 476 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00002.avro

-rw-r--r-- 1 root hdfs 478 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00003.avro

Create a Hive external table to access Avro files:

use orders;

drop table customer ;

CREATE EXTERNAL TABLE customer

(

cust\_id INT ,

cust\_name STRING ,

cust\_city STRING

)

STORED AS AVRO

location '/user/sqoop\_data/avro/';

Verify the Hive customer table:

hive> select \* from customer;

OK

1 Sam James Austin

2 Peter Carter Denver

3 Doug Smith Sunnyvale

4 Harry Warner Palo Alto

Time taken: 0.143 seconds, Fetched: 4 row(s)

hive>

Perfect! We have no errors. We successfully imported the source customer table to the target Hive table using Avro serialization. Now, we add one column to the source table and import it again to verify that we can access the target Hive table without any schema changes. Connect to MySQL and add one more column:

mysql -u sales -p

mysql>

ALTER TABLE customer ADD COLUMN cust\_state VARCHAR(15) NOT NULL;

mysql> desc customer;

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

| Field | Type | Null | Key | Default | Extra |

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

| cust\_id | int(11) | NO | PRWE | 0 | |

| cust\_name | varchar(20) | NO | | NULL | |

| cust\_city | varchar(20) | NO | | NULL | |

| CUST\_STATE | varchar(15) | YES | | NULL | |

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

4 rows in set (0.01 sec)

mysql>

Now insert rows:

INSERT into customer (cust\_id,cust\_name,cust\_city,cust\_state) values (5,'Mark Slogan','Huston','TX');

INSERT into customer (cust\_id,cust\_name,cust\_city,cust\_state) values (6,'Jane Miller','Foster City','CA');

On Hadoop, run the following sqoop command to import the customer table so as to append the new address column and data. WE have used the append and where "cust\_id > 4" parameters to import only the new rows:

sqoop import -Dmapreduce.job.user.classpath.first=true

--connect jdbc:mysql://localhost:3306/orders

--driver com.mysql.jdbc.Driver

--username sales --password xxx

--table customer

--append

--target-dir /user/sqoop\_data/avro

--as-avrodatafile

--where "cust\_id > 4"

Verify the HDFS folder:

[root@sandbox ~]# hadoop fs -ls /user/sqoop\_data/avro

Found 7 items

-rw-r--r-- 1 root hdfs 0 2017-09-09 08:57 /user/sqoop\_data/avro/\_SUCCESS

-rw-r--r-- 1 root hdfs 472 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00000.avro

-rw-r--r-- 1 root hdfs 475 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00001.avro

-rw-r--r-- 1 root hdfs 476 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00002.avro

-rw-r--r-- 1 root hdfs 478 2017-09-09 08:57 /user/sqoop\_data/avro/part-m-00003.avro

-rw-r--r-- 1 root hdfs 581 2017-09-09 09:00 /user/sqoop\_data/avro/part-m-00004.avro

-rw-r--r-- 1 root hdfs 586 2017-09-09 09:00 /user/sqoop\_data/avro/part-m-00005.avro

Now, let's verify that our target Hive table is still able to access old and new Avro files:

hive> select \* from customer;

OK

1 Sam James Austin

2 Peter Carter Denver

3 Doug Smith Sunnyvale

4 Harry Warner Palo Alto

Time taken: 0.143 seconds, Fetched: 4 row(s

Great! No errors. Still, it's business as usual; now we will add one new column to the Hive table to see the newly added Avro files:

hive> use orders;

hive> ALTER TABLE customer ADD COLUMNS (cust\_state STRING);

hive> desc customer;

OK

cust\_id int

cust\_name string

cust\_city string

cust\_state string

Time taken: 0.488 seconds, Fetched: 4 row(s

Verify the Hive table for new data:

hive> select \* from customer;

OK

1 Sam James Austin NULL

2 Peter Carter Denver NULL

3 Doug Smith Sunnyvale NULL

4 Harry Warner Palo Alto NULL

5 Mark Slogan Huston TX

6 Jane Miller Foster City CA

Time taken: 0.144 seconds, Fetched: 6 row(s)

Awesome! Take a look at customer IDs 5 and 6. We can see the newly added column (cust\_state) with values.