Neo4j - Kafka - MySQL: Configuration - Part 2

In Part 1, we configured Neo4j, Kafka and MySQL to talk using the Neo4j Kafka plugin and Maxwell's Daemon. In part 2, I will show how data can be added into MySQL and then added/modified/deleted in Neo4j through the Kafka connector.

For our dataset, I chose the Musicbrainz dataset. This dataset has several tables and has a good amount of data to test with. For this test, I am only going to use the Label and the Artists tables but you could easily add more tables and more data.

For the Label table, here is the MySQL schema.

```
CREATE TABLE `label` (   `id` bigint(20) DEFAULT NULL,   `gid` varchar(10
0) DEFAULT NULL,   `name` varchar(200) DEFAULT NULL,   `sort_name` varcha
r(200) DEFAULT NULL,   `type` int(11) DEFAULT NULL ) ENGINE=InnoDB DEFAUL
T CHARSET=latin1;
```

When we start Maxwell's Daemon, it automatically creates two topics on our Kafka machine. One is musicbrainz_musicbrainz_artist and the other is musicbrainz_musicbrainz_label.

When we do a CRUD operation on the MySQL table, the maxwell-daemon will write the operation type to the Kafka topic. For example, here is what an insert into the Artist table looks like:

```
{"database":"musicbrainz","table":"artist","type":"insert","ts":156319267
8,"xid":13695,"xoffset":87,"data":{"gid":"174442ec-e2ac-451a-a9d5-9a0669f
a2edd","name":"Goldcard","sort_name":"Goldcard","id":50419}} {"database":
"musicbrainz","table":"artist","type":"insert","ts":1563192678,"xid":1369
```

5,"xoffset":88,"data":{"gid":"2b99cd8e-55de-4a42-9cb8-6489f3195a4b","name":"Banda Black Rio","sort_name":"Banda Black Rio","id":106851}} {"database":"musicbrainz","table":"artist","type":"insert","ts":1563192678,"xid":13695,"xoffset":89,"data":{"gid":"38927bad-687f-4189-8dcf-edf1b8f716b4","name":"Kauri Kallas","sort_name":"Kallas, Kauri","id":883445}}

The Neo4j cypher statement in our neo4j.conf file will read from that topic, determine the operation (insert, update or delete) and modify data in Neo4j appropriately.

```
streams.sink.topic.cypher.musicbrainz_musicbrainz_artist=FOREACH(ignoreMe
 IN CASE WHEN event.type='insert' THEN [1] ELSE [] END | MERGE (u:Artist{
gid:event.data.gid}) on match set u.id = event.data.id, u.name=event.data
.name, u.sort_name=event.data.sort_name on create set u.id = event.data.i
d, u.name=event.data.name, u.sort_name=event.data.sort_name) FOREACH(igno
reMe IN CASE WHEN event.type='delete' THEN [1] ELSE [] END | MERGE (u:Art
ist{gid:event.data.gid})  detach delete u) FOREACH(ignoreMe IN CASE WHEN
event.type='update' THEN [1] ELSE [] END | MERGE (u:Artist{gid:event.data
.gid})  set u.id = event.data.id, u.name=event.data.name, u.sort_name=eve
nt.data.sort_name)
```

To show how this works, we will remove a Label and then re-add that same label. In our Neo4j database, we already have 163K labels. We will delete the XTOY label from MySQL and watch the delete get placed on the Kafka queue and then removed from Neo4j.

In Neo4j, here is the XTOY label.

In MySQL, we are going to run:

```
delete from label where name='XTOY'
```

In our Kafka musicbrainz_musicbrainz_label topic, we see a delete statement coming from MySQL:


```
{"database":"musicbrainz","table":"label","type":"delete","ts":1563197724
,"xid":22983,"commit":true,"data":{"id":27894,"gid":"e8c1f93b-f518-43f2-b
9be-e00e31a5e92d","name":"XTOY","sort_name":"2000","type":-1}}
```


Neo4j polls the topic, evaluates the type of action and acts appropriately. Int his case, it should delete the XTOY label. Let's look at Neo4j now and see if the XTOY label has been removed:

We see that it has been removed. Now, let's reinsert the record into MySQL and see if Neo4j picks up the insert.

```
INSERT INTO label(id,gid,name, sort_name, type) values(27894,'e8c1f93b-f5
18-43f2-b9be-e00e31a5e92d','XTOY','XTOY',-1)  {"database":"musicbrainz","
table":"label","type":"insert","ts":1563197966,"xid":23309,"commit":true,
"data":{"id":27894,"gid":"e8c1f93b-f518-43f2-b9be-
e00e31a5e92d","name":"XTOY","sort_name":"XTOY","type":-1}}
```

Checking Neo4j once more we see the node has been added in.

The data is automatically replicated across the Neo4j cluster so we don't have to worry about that aspect.

In summary, it is straight-forward to sync database changes from MySQL to Neo4j through Kafka.