**Data Migration Using Spark**

**1). Spark Command Prompt**

Step1: Download and Install Spark (spark-2.4.5-bin-hadoop2.7)

Step2: For Connecting Spark with Postgres and Cassandra

* Download postgresql-42.2.6.jar
* Download spark-cassandra-connector-2.4.0-s\_2.11.jar
* Place the jar files in jars folder of spark

Step3: Start the Spark Shell using the command (Go to Spark bin path)

Priys-MacBook-Air:bin priyranjan$ ./spark-shell --conf spark.driver.host=localhost --conf spark.cassandra.connection.host=127.0.0.1

Step4: Set the Properties for connection with Postges using the command

scala> import java.util.Properties

scala> val connectionProperties = new Properties()

scala> connectionProperties.setProperty("user", "postgres")

scala> connectionProperties.setProperty("password", "postgres")

scala> connectionProperties.setProperty("Driver", "org.postgresql.Driver")

scala> val url = "jdbc:postgresql://localhost:5432/CustomerData"

Step5:Write Query to get data from employee table in Postgres

scala>val query1 = "(SELECT \* FROM employee) as e1"

Step6:Create data frame

scala>val query1df = spark.read.jdbc(url, query1, connectionProperties)

Step7:Show data of the dataframe

scala>query1df.show()

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

| id| email|first\_name|last\_name|

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

| 1| first@mail.com| priy| ranjan|

| 2|second@mail.com| ram| kumar|

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

Step8:Write the data in Cassandra (table -> employee and keyspace -> demo)

Open Cassandra cqlsh

* Create Keyspace in Cassandra

CREATE KEYSPACE demo WITH replication = {'class':'SimpleStrategy', 'replication\_factor' : 1};

* Create Employee table in Cassandra

CREATE TABLE employee (id int PRIMARY KEY, email text, first\_name text, last\_name text );

scala>query1df.write.format("org.apache.spark.sql.cassandra").options(Map("table" -> "employee", "keyspace" -> "demo")).save()

Step9:Using the above steps data is transferred to Cassandra.

Open cqlsh and check.

cqlsh:demo> select \* from employee;

**id** | **email** | **first\_name** | **last\_name**

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

**1** | **first@mail.com** | **priy** | **ranjan**

**2** | **second@mail.com** | **ram** | **kumar**

2). Spark Java

Step1: Create a maven project

Step2: Add dependency for Spark ,Postgres and Cassandra

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>9.4-1201-jdbc41</version>

</dependency>

<!--Spark Cassandra Connector -->

<dependency>

<groupId>com.datastax.spark</groupId>

<artifactId>spark-cassandra-connector\_2.11</artifactId>

<version>2.0.5</version>

</dependency>

<dependency>

<groupId>com.datastax.spark</groupId>

<artifactId>spark-cassandra-connector-java\_2.11</artifactId>

<version>1.5.2</version>

</dependency>

<!--Spark -->

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-core\_2.11</artifactId>

<version>2.2.0</version>

</dependency>

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-streaming\_2.11</artifactId>

<version>2.2.0</version>

</dependency>

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-sql\_2.11</artifactId>

<version>2.2.0</version>

</dependency>

Step3:Create tables in postgres employee and account having onetomany relation

Employee:

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

| id| email|first\_name|last\_name|

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

| 1| first@mail.com| priy| ranjan|

| 2|second@mail.com| ram| kumar|

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

Account:

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

| id|acc\_number|employee\_id|

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

| 1| 123| 1|

| 2| 456| 1|

| 3| 789| 2|

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

Step4: Write Java Class to fetch data from Postgres and insert in Cassandra using Sql

* Create table employee and account in Cassandra

CREATE TABLE employee (id int PRIMARY KEY, email text, first\_name text, last\_name text );

CREATE TABLE account (id int PRIMARY KEY, acc\_number text, employee\_id int );

**public** **class** DataTransfer {

**public** **static** **void** main(String[] args) {

/\* create SaprkConf and set the properties \*/

SparkConf conf= **new** SparkConf();

conf.setAppName("Java Spark");

conf.setMaster("local[\*]");

conf.set("spark.cassandra.connection.host", "127.0.0.1");

/\* create SparkSession \*/

SparkSession spark = SparkSession

.*builder*()

.config(conf)

.getOrCreate();

/\* Query to get data from Postgres Employee and Account table \*/

String query = "(SELECT \* FROM employee) as e";

String query1 = "(SELECT \* FROM account) as a";

/\* Url for Postgres Connection \*/

String url = "jdbc:postgresql://localhost:5432/CustomerData";

/\* Read data from Postgres using query from Employee table\*/

Dataset<Row> df = spark

.read()

.format( "jdbc" )

.option( "url", url )

.option( "dbtable", query )

.option( "password", "postgres" )

.option( "user", "postgres" )

.load();

df.show();

/\* Write data to Cassandra Employee Table \*/

df.write().format("org.apache.spark.sql.cassandra").option("table", "employee").option("keyspace", "demo").save();

/\* Read data from Postgres using query1 from Account table\*/

Dataset<Row> df1 = spark

.read()

.format( "jdbc" )

.option( "url", url )

.option( "dbtable", query1 )

.option( "password", "postgres" )

.option( "user", "postgres" )

.load();

df1.show();

/\* Write data to Cassandra account Table \*/

df1.write().format("org.apache.spark.sql.cassandra").option("table", "account").option("keyspace", "demo").save();

}

}

Check in Cassandra Cqlsh:

cqlsh:demo> select \* from employee;

**id** | **email** | **first\_name** | **last\_name**

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

**1** | **first@mail.com** | **priy** | **ranjan**

**2** | **second@mail.com** | **ram** | **kumar**

cqlsh:demo> select \* from account;

**id** | **acc\_number** | **employee\_id**

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

**1** | **123** | **1**

**2** | **456** | **1**

**3** | **789** | **2**

Step5: Write Java Class to fetch data from Postgres and insert in Cassandra using Cassandra Java Function

Create Employee Class

**public** **class** Employee **implements** Serializable{

**private** Integer id;

**private** String email;

**private** String first\_name;

**private** String last\_name;

// getter setter

}

**public** **class** DataTransfer1 {

**public** **static** **void** main(String[] args) {

SparkConf conf = **new** SparkConf();

conf.setAppName("Java Spark");

conf.setMaster("local[\*]");

conf.set("spark.cassandra.connection.host", "127.0.0.1");

SparkSession spark = SparkSession.*builder*().config(conf).getOrCreate();

String query = "(SELECT \* FROM employee) as e1";

String url = "jdbc:postgresql://localhost:5432/CustomerData";

Dataset<Row> df = spark.read().format("jdbc").option("url", url).option("dbtable", query)

.option("password", "postgres").option("user", "postgres").load();

df.show();

Dataset<Employee> df1 = df.as(Encoders.*bean*(Employee.**class**));

List<Employee> rowslist = df1.collectAsList();

JavaSparkContext javaSparkContext = **new** JavaSparkContext(spark.sparkContext());

JavaRDD<Employee> empRDD = javaSparkContext.parallelize(rowslist);

CassandraJavaUtil.*javaFunctions*(empRDD).writerBuilder("demo", "employee", CassandraJavaUtil.*mapToRow*(Employee.**class**)).saveToCassandra();

javaSparkContext.close();

spark.close();

}

}

Step6: Write Java Class to fetch data from Postgres relational data and Map according to Cassandra table and insert in Cassandra using Cassandra Java Function

* Create class EmployeeEntity

**public** **class** EmployeeEntity **implements** Serializable {

**private** Integer id;

**private** String email;

**private** String first\_name;

**private** String last\_name;

**private** Integer acnt\_id;

**private** String acc\_number;

**private** Integer employee\_id;

//getter setters

}

* Create type accountdetail and table employeedetail in Cassandra for mapping the postgres relation in Cassandra

CREATE TYPE accountdetail ( acnt\_id int , acc\_number text, employee\_id int);

CREATE TABLE employeedetail ( id int PRIMARY KEY, email text, first\_name text, last\_name text ,accounts list<frozen<accountdetail>>);

* Create class EmployeeDetail and AccountDetail

**public** **class** EmployeeDetail **implements** Serializable{

**private** Integer id;

**private** String email;

**private** String first\_name;

**private** String last\_name;

**private** List<AccountDetail> accounts;

**//getter setters**

**}**

**public** **class** AccountDetail **implements** Serializable{

**private** Integer acnt\_id;

**private** String acc\_number;

**private** Integer employee\_id;

**//getter setters**

}

**public** **class** DataMapping {

**public** **static** **void** main(String[] args) {

SparkConf conf = **new** SparkConf();

conf.setAppName("Java Spark");

conf.setMaster("local[\*]");

conf.set("spark.cassandra.connection.host", "127.0.0.1");

SparkSession spark = SparkSession.*builder*().config(conf).getOrCreate();

/\* Query to get employee and their account from Postgres \*/

String query = " (select e1.\*,a1.acc\_number,a1.id as acnt\_id,a1.employee\_id from employee e1 join account a1 on e1.id=a1.employee\_id) as a";

String url = "jdbc:postgresql://localhost:5432/CustomerData";

Dataset<Row> df = spark.read().format("jdbc").option("url", url).option("dbtable", query)

.option("password", "postgres").option("user", "postgres").load();

df.show();

/\* Map the result from postgres according to Cassandra Model (EmployeeDetail having list of AccountDetail ) \*/

Dataset<EmployeeEntity> df1 = df.as(Encoders.*bean*(EmployeeEntity.**class**));

List<EmployeeEntity> rowslist = df1.collectAsList();

List<EmployeeDetail> list = **new** ArrayList<EmployeeDetail>();

HashMap<Integer, List<EmployeeEntity>> hm = **new** HashMap<Integer, List<EmployeeEntity>>();

**for** (EmployeeEntity emp : rowslist) {

**if** (hm.get(emp.getId()) != **null**) {

List<EmployeeEntity> ls = hm.get(emp.getId());

ls.add(emp);

hm.put(emp.getId(), ls);

} **else** {

List<EmployeeEntity> ls = **new** ArrayList<EmployeeEntity>();

ls.add(emp);

hm.put(emp.getId(), ls);

}

}

**for** (Integer i : hm.keySet()) {

List<EmployeeEntity> emps = hm.get(i);

EmployeeDetail e1 = **new** EmployeeDetail();

e1.setId(emps.get(0).getId());

e1.setEmail(emps.get(0).getEmail());

e1.setFirst\_name(emps.get(0).getFirst\_name());

e1.setLast\_name(emps.get(0).getLast\_name());

List<AccountDetail> acntList = **new** ArrayList<AccountDetail>();

emps.forEach(e -> {

AccountDetail ac = **new** AccountDetail();

ac.setAcnt\_id(e.getAcnt\_id());

ac.setAcc\_number(e.getAcc\_number());

ac.setEmployee\_id(e.getEmployee\_id());

acntList.add(ac);

});

e1.setAccounts(acntList);

list.add(e1);

}

JavaSparkContext javaSparkContext = **new** JavaSparkContext(spark.sparkContext());

JavaRDD<EmployeeDetail> userRDD = javaSparkContext.parallelize(list);

/\* Save the Mapped EmployeeDetail in Cassandra table \*/

CassandraJavaUtil.*javaFunctions*(userRDD)

.writerBuilder("demo", "employeedetail", CassandraJavaUtil.*mapToRow*(EmployeeDetail.**class**)).saveToCassandra();

javaSparkContext.close();

}

}

* Open Cqlsh and verify
* cqlsh:demo> select \* from employeedetail;

**id** | **accounts** | **email** | **first\_name** | **last\_name**

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

**1** | **[{acnt\_id: 1, acc\_number: '123', employee\_id: 1}, {acnt\_id: 2, acc\_number: '456', employee\_id: 1}]** | **first@mail.com** | **priy** | **ranjan**

**2** | **[{acnt\_id: 3, acc\_number: '789', employee\_id: 2}]** | **second@mail.com** | **ram** | **kumar**

cqlsh:demo> select JSON \* from employeedetail;

**[json]**

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**{"id": 1, "accounts": [{"acnt\_id": 1, "acc\_number": "123", "employee\_id": 1}, {"acnt\_id": 2, "acc\_number": "456", "employee\_id": 1}], "email": "first@mail.com", "first\_name": "priy", "last\_name": "ranjan"}**

**{"id": 2, "accounts": [{"acnt\_id": 3, "acc\_number": "789", "employee\_id": 2}], "email": "second@mail.com", "first\_name": "ram", "last\_name": "kumar"}**

Step7:Fetch data From Cassandra tables and join Using Cassandra Java Function

* Create class Employee and Account

**public** **class** Account **implements** Serializable{

**private** Integer id;

**private** String acc\_number;

**private** Integer employee\_id;

}

**public** **class** Employee **implements** Serializable{

**private** Integer id;

**private** String email;

**private** String first\_name;

**private** String last\_name;

}

* Create class to get Employee and Account data from Cassandra tables and join the result

**public** **class** DataTransfer3 {

**private** **static** **void** showResults(JavaSparkContext sc) {

/\* Get records from employee table \*/

JavaPairRDD<Integer, Employee> empRdd = *javaFunctions*(sc)

.cassandraTable("demo", "employee", *mapRowTo*(Employee.**class**))

.keyBy(**new** Function<Employee, Integer>() {

@Override

**public** Integer call(Employee emp) **throws** Exception {

**return** emp.getId();

}

});

/\* Get records from account table \*/

JavaPairRDD<Integer, Account> accntRdd = *javaFunctions*(sc)

.cassandraTable("demo", "account", *mapRowTo*(Account.**class**))

.keyBy(**new** Function<Account, Integer>() {

@Override

**public** Integer call(Account acc) **throws** Exception {

**return** acc.getEmployee\_id();

}

});

/\* Join Employee and Account based on employee id \*/

List<Tuple2<Employee, Optional<Account>>> results = empRdd.leftOuterJoin(accntRdd).values().collect();

**for** (Tuple2<Employee, Optional<Account>> result : results) {

System.***out***.println(result);

}

}

**public** **static** **void** main(String[] args) {

SparkConf conf = **new** SparkConf();

conf.setAppName("Java API demo");

conf.setMaster("local[\*]");

conf.set("spark.cassandra.connection.host", "127.0.0.1");

JavaSparkContext sc = **new** JavaSparkContext(conf);

*showResults*(sc);

sc.stop();

}

}

Result:

(Employee{id=1, email='first@mail.com', firstName='priy' ,lastName=ranjan},Optional[Account{id=1, accountNumber='123',employeeId=1}])

(Employee{id=1, email='first@mail.com', firstName='priy' ,lastName=ranjan},Optional[Account{id=2, accountNumber='456',employeeId=1}])

(Employee{id=2, email='second@mail.com', firstName='ram' ,lastName=kumar},Optional[Account{id=3, accountNumber='789',employeeId=2}])