组件选型，优化点评审！

Kafka持久化至Hive，目前搜到大致有如下几种方案：

1、HiveKa : Apache Hive's storage handler that adds support in Apache Hive to query data from Apache Kafka

https://github.com/HiveKa/HiveKa

2、Confluent Platform - HDFS Connector

http://kaimingwan.com/post/kafka/kafkachi-jiu-hua-shu-ju-dao-hdfsde-fang-fa

http://docs.confluent.io/2.0.0/connect/connect-hdfs/docs/index.html

3、camus或gobblin

http://www.aboutyun.com/thread-20701-1-1.html

oracle数据库字段配置:

user table\_ name column\_name primary\_key

ood table1 c1,c2,c3,c4…… c1,c2,c3,c4……

场景概要：goldengate监控oracle数据数据库，数据发生变化时，发送信息到kafka，用spark streaming消费kafka数据后，入库hive或者hbase！

Kafka消息示列：

DEBUG:partition : 0, offset : 210 mess : {"table":"BDTEST.TEST\_KPI\_SCORE","op\_type":"I","op\_ts":"2016-11-09 07:16:57.385885","current\_ts":"2016-11-10T17:29:12.479000","pos":"00000000-10000001391","after":{"SCORE\_ID":0,"KPI\_CYCLE":"4","EVALUATE\_ID":"MM\_20\_04\_03","DEPT\_CODE":"0302","P\_ORGANIZATION\_CODE":null,"YM":"200912","NUMERATOR":"0","DENOMINATOR":"0","SCORE":"0","SUM\_SCORE":null,"STATUS":null,"ETL\_DT":"20120814151900"}}

解析数据与入库流程：

1 解析kafka数据

获取table（表名称）op\_type（增删改查的操作类型） after （具体的字段）op\_ts(操作时间)

业务涵盖点：增加和修改需匹配oracle数据库中的字段名称 进行入库和修改操作

删除 与查询不需匹配相关信息

代码点：解析这串json信息 获取oracle表中的数据

2 对数据字段进行匹配后，做数据库入库hive操作

代码点：做入库操作

测试表：

DIM\_LCAM\_PROJECT\_TYPE,SM\_PRIV\_MENU\_USER

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

| OWNER| TABLE\_NAME| TABLE\_COLS| TABLE\_PK|

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

| KIT\_DEV|DIM\_LCAM\_PROJECT\_...|PROJECT\_TYPE\_ID,P...|PROJECT\_TYPE\_ID,P...|

| KIT\_DEV| SM\_PRIV\_MENU\_USER|ID,EIPID,MENU\_ID,...| ID|

|KIT\_DEV1| test| id,pwd,name| id|

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

|  |  |  |  |
| --- | --- | --- | --- |
| OWNER | TABLE\_NAME | TABLE\_COLS | TABLE\_PK |
| KIT\_DEV | DIM\_LCAM\_PROJECT\_TYPE | PROJECT\_TYPE\_ID,PROJECT\_TYPE\_NAME,UNIFIED\_CODE,SORT | PROJECT\_TYPE\_ID,PROJECT\_TYPE\_NAME,UNIFIED\_CODE,SORT |
| KIT\_DEV | SM\_PRIV\_MENU\_USER | ID,EIPID,MENU\_ID,UPD\_USER,UPD\_DATE,CRT\_USER,CRT\_DATE | ID |

Hive 语句建表: DIM\_LCAM\_PROJECT\_TYPE 表

drop table if exists DIM\_LCAM\_PROJECT\_TYPE;

create table DIM\_LCAM\_PROJECT\_TYPE (PROJECT\_TYPE\_ID varchar(100), PROJECT\_TYPE\_NAME varchar(100), UNIFIED\_CODE varchar(100), SORT varchar(100)) clustered by (PROJECT\_TYPE\_ID) into 10 buckets stored as orc TBLProperties(\"transactional\"=\"true\");

Hive 语句建表: SM\_PRIV\_MENU\_USER 表

drop table if exists SM\_PRIV\_MENU\_USER;

create table SM\_PRIV\_MENU\_USER (ID varchar(100), EIPID varchar(100), MENU\_ID varchar(100), UPD\_USER varchar(100),

UPD\_DATE varchar(100), CRT\_USER varchar(100), CRT\_DATE varchar(100)) clustered by (ID) into 10 buckets stored as orc TBLProperties(\"transactional\"=\"true\")

测试脚本：

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519501', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519502', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519503', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519504', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519505', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

INSERT INTO "KIT\_DEV"."SM\_PRIV\_MENU\_USER" (ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE) VALUES

('201701162519506', 'zhangwenfeng@gsbb.gpgc1', '23', 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'), 'admin', TO\_DATE('17-4月 -12', 'DD-MON-RR'));

commit;

update SM\_PRIV\_MENU\_USER set UPD\_USER='admin123' where id='201701162519501';

update SM\_PRIV\_MENU\_USER set UPD\_USER='admin1213' where id='201701162519502';

commit;

delete SM\_PRIV\_MENU\_USER where id='201701162519501';

delete SM\_PRIV\_MENU\_USER where id='201701162519502';

delete SM\_PRIV\_MENU\_USER where id='201701162519503';

delete SM\_PRIV\_MENU\_USER where id='201701162519504';

delete SM\_PRIV\_MENU\_USER where id='201701162519505';

delete SM\_PRIV\_MENU\_USER where id='201701162519506';

commit;

hive表操作：truncate table SM\_PRIV\_MENU\_USER;

Q:

Hive下的重复数据如何分辨识别

单引号下的注入式攻击

关于分桶字段不可更改的问题:

String sql5 = "create table DIM\_LCAM\_PROJECT\_TYPE (PROJECT\_TYPE\_ID varchar(100), PROJECT\_TYPE\_NAME varchar(100), UNIFIED\_CODE varchar(100), SORT varchar(100)) clustered by (PROJECT\_TYPE\_ID) into 10 buckets stored as orc TBLProperties(\"transactional\"=\"true\")";

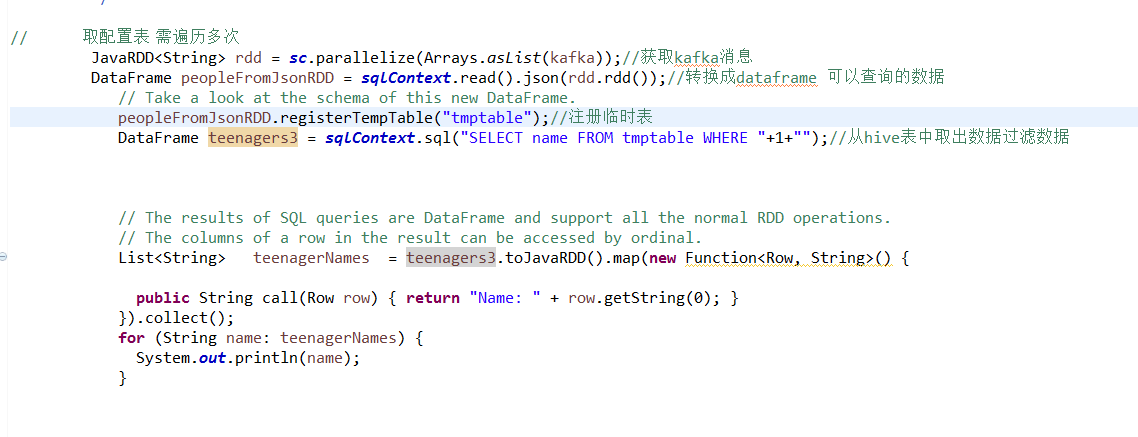


Kafka的消费入库hive先告一段落,接下来测试关联查询！

遍历kafka消息后入库不同的表：

设计思路：

目前已实现了表的增删改查操作，目前在入库之前需要做关联对比操作！



步骤1： 根据表名 取出需过滤的条件

步骤2： 封装json为rdd

步骤3：注册临时表

步骤4： 获取集合数据

步骤5 遍历数据

步骤6 如果匹配相关规则 则入库 没有数据则过滤掉

Rule :

select \*from kitdev.rule\_cfg

rule\_id | rule\_name | rule\_desc | rule\_type | rule\_sql | system\_name | table\_name | column\_name | rule\_status

过滤sql：

select rule\_sql from kitdev.rule\_cfg where table\_name=’’;

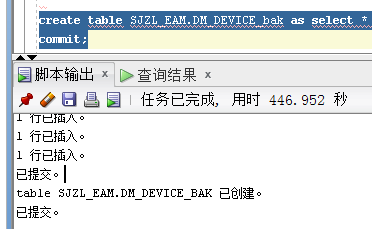
get the result:

Row[] lengths = jdbcDF.collect();

**for** (Row row : lengths) {

System.***out***.println(row.get(0));

}



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

| CREATE TABLE kitdev.rule\_cfg( |

| rule\_id string DEFAULT NULL COMMENT ' ', |

| rule\_code string DEFAULT NULL COMMENT ' ', |

| rule\_name string DEFAULT NULL COMMENT ' ', |

| rule\_desc string DEFAULT NULL COMMENT ' ', |

| rule\_type string DEFAULT NULL COMMENT ' ', |

| rule\_range\_col\_name string DEFAULT NULL COMMENT ' ', |

| rule\_range\_col\_value string DEFAULT NULL COMMENT ' ', |

| rule\_status string DEFAULT NULL COMMENT ' ', |

| src\_sys\_code string DEFAULT NULL COMMENT ' ', |

| table\_name string DEFAULT NULL COMMENT ' ', |

| col\_name string DEFAULT NULL COMMENT ' ', |

| rule\_sql string DEFAULT NULL COMMENT ' ' |

| )

修改后的代码为：

1. 根据表名获取cols字段
2. 根据cols字段获取kafka过来的值
3. 根据表名以及步骤2获取的kafka的值 获取rule sql 进行

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

| CREATE TABLE RULE\_CHECK\_RESULT\_TODAY( |

| rule\_id string DEFAULT NULL COMMENT ' ', |

| table\_name string DEFAULT NULL COMMENT ' ', |

| table\_pk\_value string DEFAULT NULL COMMENT ' ', |

| rule\_check\_time string DEFAULT NULL COMMENT ' ', |

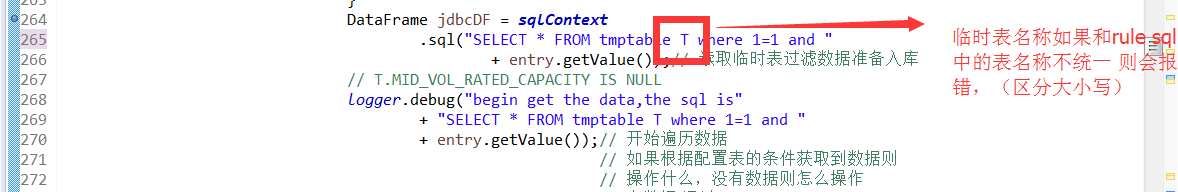
| check\_result string DEFAULT NULL COMMENT ' ' |

| )

DM\_A\_TRANSFORMER

SJZL\_EAM.DM\_A\_BREAKER;

Q1:

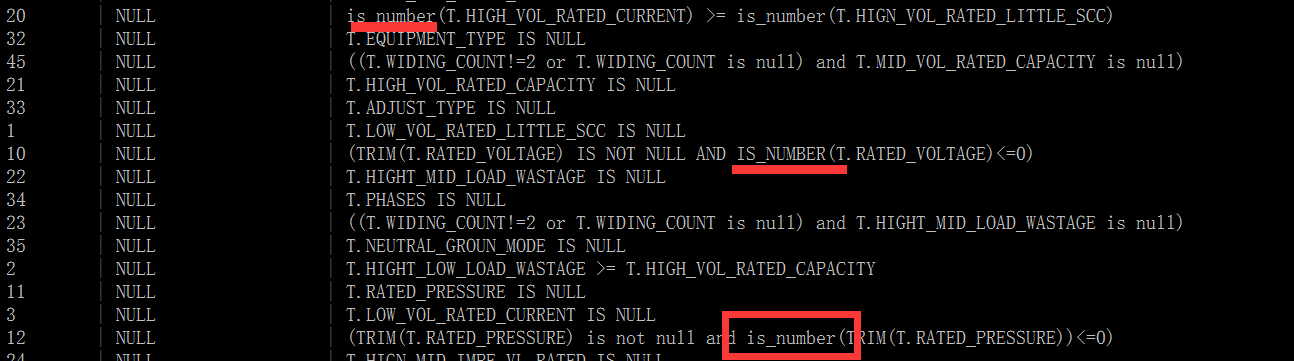


org.apache.spark.sql.AnalysisException: cannot resolve 'T.MID\_VOL\_RATED\_CAPACITY' given input columns INDUSTRIAL\_CAPACITY, HIGH\_VOLTAGE\_BOX\_GAS, RATED\_PRESSURE, HIGH\_VOL\_RATED\_CAPACITY, ENVELOP\_MODE, UPPER\_OIL\_BOX\_WEIGHT, PROVINCE\_CODE, MID\_LOW\_IMPEDANCE\_VL, EQUIPMENT\_TYPE, LINE\_NAME, RESIDENTIAL\_CAPACITY, TAP\_AREA, CUSTOMER\_NO, IMPEDANCE\_VOLTAGE, DYRCP, TRANSFORMER\_NO, HIGN\_LOW\_IMPE\_VL\_RATED, OIL\_BOX\_AIRPROOF\_MODE, BURIED\_DIMENSIONS, WIDING\_COUNT, LATITUDE, ADJUST\_TYPE, WINDING\_TEMP\_ALARM\_VALUE, HIGN\_VOL\_RATED\_LITTLE\_SCC, NATURE\_OF\_ELE\_CONSUMED, COMMERCIAL\_CAPACITY, HIGN\_MID\_IMPE\_VL\_MIN, LONGITUDE, HIGHT\_MID\_LOAD\_WASTAGE, BUREAU\_CODE, MID\_VOL\_RATED\_LITTLE\_SCC, MACHINE\_WEIGHT, LOW\_VOL\_RATED\_LITTLE\_SCC, PHASES, BULK\_GAS, HIGN\_MID\_IMPE\_VL\_RATED, OIL\_MANUFACTURER, MID\_LOW\_LOAD\_WASTAGE, WINDING\_MATERIAL, NON\_INDUSTRIAL\_CAPACITY, OLTC\_GAS, NO\_LOAD\_WASTAGE, IS\_STANDBY\_PHASE, RATED\_VOLTAGE, NEUTRAL\_GROUN\_MODE, MID\_VOL\_RATED\_CAPACITY, HIGH\_VOL\_RATED\_CURRENT, RATED\_CAPACITY, FARMLAND\_CAPACITY, LINE\_NO, LOAD\_LOSS, IRON\_CORE\_MODE, NO\_LOAD\_CURRENT, DATA\_FROM, TRAFFIC\_WEIGHT, COOLING\_SYSTEM\_GAS, HIGHT\_LOW\_LOAD\_WASTAGE, ID, UPDATE\_TIME, STANDARD\_CODE, LOW\_VOL\_RATED\_CURRENT, RATED\_VOLTAGE\_RATIO, OPTIMISTIC\_LOCK\_VERSION, CUSTOMER\_NAME, HIGN\_LOW\_IMPE\_VL\_MAX, TAP\_POSITION, INSTALLATION\_TYPE, TIE\_LINE\_GROUP, OIL\_TEMP\_ALARM\_VALUE, INSULATION\_LEVEL, TYPE, OIL\_MARK, OIL\_WEIGHT, INSULATION\_CORROSION\_LEVEL, YAWP\_VALUE, LOW\_VOL\_RATED\_CAPACITY, HIGN\_MID\_IMPE\_VL\_MAX, HIGN\_LOW\_IMPE\_VL\_MIN, MID\_VOL\_RATED\_CURRENT, ZERO\_SEQ\_IMPEDANCE, COOLING\_MODE, TOTAL\_WEIGHT;

Q2:

在做数据过滤的时候，做规则匹配的时候 ，发现有些函数在spark中识别不了，会报错：

自定义函数：



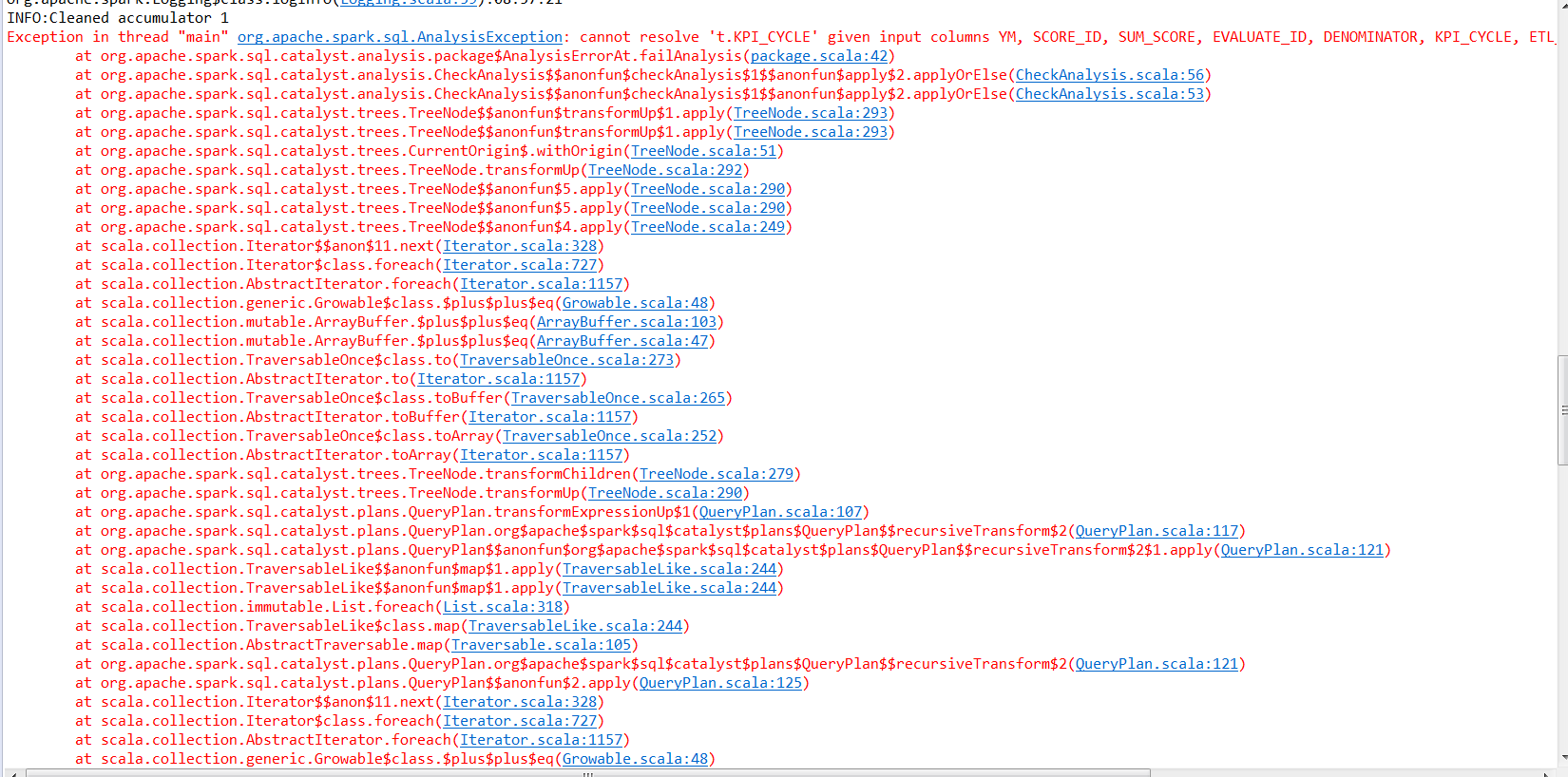
org.apache.spark.sql.AnalysisException: undefined function IS\_NUMBER;

结果表：RULE\_CHECK\_RESULT\_TODAY

// sqlContext.sql(sql);

// jdbcDF.toDF().insertInto("tableName");这两种都是数据入库的操作方式 年后回来进行测试 部署到集群

Todolist： 1 反馈自定义函数问题给辉哥 以及临时表的大小写问题



2 测试spark sql方式 而不是重复的连接hive数据库

验证临时表的rdd对象与真正的的hive表 是否需要与字段匹配

worldBankDf.toDF().~~insertInto~~("tableName");

批量测试语句：

declare

begin

for i in 1 .. 5 loop

insert into SM\_PRIV\_MENU\_USER

(ID, EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE)

select to\_number('99998'||i), EIPID, MENU\_ID, UPD\_USER, UPD\_DATE, CRT\_USER, CRT\_DATE from SM\_PRIV\_MENU\_USER t where rownum = 1;

commit;

end loop;

end;

规则表相关：批量添加

declare

begin

for i in 1 .. 50 loop

insert into DM\_A\_TRANSFORMER

(ID,BUREAU\_CODE,TYPE,HIGN\_LOW\_IMPE\_VL\_MAX,HIGN\_LOW\_IMPE\_VL\_RATED,HIGN\_LOW\_IMPE\_VL\_MIN,HIGN\_MID\_IMPE\_VL\_MAX,HIGN\_MID\_IMPE\_VL\_RATED,HIGN\_MID\_IMPE\_VL\_MIN,MID\_LOW\_IMPEDANCE\_VL,NO\_LOAD\_WASTAGE,NO\_LOAD\_CURRENT,ZERO\_SEQ\_IMPEDANCE,MID\_VOL\_RATED\_LITTLE\_SCC,LOW\_VOL\_RATED\_LITTLE\_SCC,OIL\_WEIGHT,OIL\_MARK,TRAFFIC\_WEIGHT,UPPER\_OIL\_BOX\_WEIGHT,MACHINE\_WEIGHT,TOTAL\_WEIGHT,OIL\_MANUFACTURER,OIL\_TEMP\_ALARM\_VALUE,WINDING\_TEMP\_ALARM\_VALUE,RATED\_VOLTAGE,ADJUST\_TYPE,RATED\_VOLTAGE\_RATIO,COOLING\_MODE,HIGH\_VOL\_RATED\_CAPACITY,HIGH\_VOL\_RATED\_CURRENT,MID\_VOL\_RATED\_CAPACITY,MID\_VOL\_RATED\_CURRENT,LOW\_VOL\_RATED\_CAPACITY,LOW\_VOL\_RATED\_CURRENT,TIE\_LINE\_GROUP,INSULATION\_LEVEL,STANDARD\_CODE,HIGHT\_LOW\_LOAD\_WASTAGE,HIGHT\_MID\_LOAD\_WASTAGE,MID\_LOW\_LOAD\_WASTAGE,PHASES,DYRCP,WIDING\_COUNT,IRON\_CORE\_MODE,WINDING\_MATERIAL,INSULATION\_CORROSION\_LEVEL,OIL\_BOX\_AIRPROOF\_MODE,YAWP\_VALUE,ENVELOP\_MODE,INSTALLATION\_TYPE,TAP\_POSITION,TAP\_AREA,HIGN\_VOL\_RATED\_LITTLE\_SCC,CUSTOMER\_NO,CUSTOMER\_NAME,TRANSFORMER\_NO,LINE\_NO,LINE\_NAME,LATITUDE,LONGITUDE,BURIED\_DIMENSIONS,NATURE\_OF\_ELE\_CONSUMED,RESIDENTIAL\_CAPACITY,COMMERCIAL\_CAPACITY,INDUSTRIAL\_CAPACITY,NON\_INDUSTRIAL\_CAPACITY,FARMLAND\_CAPACITY,BULK\_GAS,RATED\_PRESSURE,RATED\_CAPACITY,LOAD\_LOSS,HIGH\_VOLTAGE\_BOX\_GAS,COOLING\_SYSTEM\_GAS,EQUIPMENT\_TYPE,IS\_STANDBY\_PHASE,OLTC\_GAS,NEUTRAL\_GROUN\_MODE,IMPEDANCE\_VOLTAGE,OPTIMISTIC\_LOCK\_VERSION,UPDATE\_TIME,DATA\_FROM,PROVINCE\_CODE)

select to\_number('20170208002'||i), BUREAU\_CODE,TYPE,HIGN\_LOW\_IMPE\_VL\_MAX,HIGN\_LOW\_IMPE\_VL\_RATED,HIGN\_LOW\_IMPE\_VL\_MIN,HIGN\_MID\_IMPE\_VL\_MAX,HIGN\_MID\_IMPE\_VL\_RATED,HIGN\_MID\_IMPE\_VL\_MIN,MID\_LOW\_IMPEDANCE\_VL,NO\_LOAD\_WASTAGE,NO\_LOAD\_CURRENT,ZERO\_SEQ\_IMPEDANCE,MID\_VOL\_RATED\_LITTLE\_SCC,LOW\_VOL\_RATED\_LITTLE\_SCC,OIL\_WEIGHT,OIL\_MARK,TRAFFIC\_WEIGHT,UPPER\_OIL\_BOX\_WEIGHT,MACHINE\_WEIGHT,TOTAL\_WEIGHT,OIL\_MANUFACTURER,OIL\_TEMP\_ALARM\_VALUE,WINDING\_TEMP\_ALARM\_VALUE,RATED\_VOLTAGE,ADJUST\_TYPE,RATED\_VOLTAGE\_RATIO,COOLING\_MODE,HIGH\_VOL\_RATED\_CAPACITY,HIGH\_VOL\_RATED\_CURRENT,MID\_VOL\_RATED\_CAPACITY,MID\_VOL\_RATED\_CURRENT,LOW\_VOL\_RATED\_CAPACITY,LOW\_VOL\_RATED\_CURRENT,TIE\_LINE\_GROUP,INSULATION\_LEVEL,STANDARD\_CODE,HIGHT\_LOW\_LOAD\_WASTAGE,HIGHT\_MID\_LOAD\_WASTAGE,MID\_LOW\_LOAD\_WASTAGE,PHASES,DYRCP,WIDING\_COUNT,IRON\_CORE\_MODE,WINDING\_MATERIAL,INSULATION\_CORROSION\_LEVEL,OIL\_BOX\_AIRPROOF\_MODE,YAWP\_VALUE,ENVELOP\_MODE,INSTALLATION\_TYPE,TAP\_POSITION,TAP\_AREA,HIGN\_VOL\_RATED\_LITTLE\_SCC,CUSTOMER\_NO,CUSTOMER\_NAME,TRANSFORMER\_NO,LINE\_NO,LINE\_NAME,LATITUDE,LONGITUDE,BURIED\_DIMENSIONS,NATURE\_OF\_ELE\_CONSUMED,RESIDENTIAL\_CAPACITY,COMMERCIAL\_CAPACITY,INDUSTRIAL\_CAPACITY,NON\_INDUSTRIAL\_CAPACITY,FARMLAND\_CAPACITY,BULK\_GAS,RATED\_PRESSURE,RATED\_CAPACITY,LOAD\_LOSS,HIGH\_VOLTAGE\_BOX\_GAS,COOLING\_SYSTEM\_GAS,EQUIPMENT\_TYPE,IS\_STANDBY\_PHASE,OLTC\_GAS,NEUTRAL\_GROUN\_MODE,IMPEDANCE\_VOLTAGE,OPTIMISTIC\_LOCK\_VERSION,UPDATE\_TIME,DATA\_FROM,PROVINCE\_CODE from DM\_A\_TRANSFORMER t where rownum = 1;

commit;

end loop;

end;

2017-02-09 报错日志:

ERROR:Error opening session

org.apache.thrift.TApplicationException: Required field 'client\_protocol' is unset! Struct:TOpenSessionReq(client\_protocol:null, configuration:{use:database=default})

at org.apache.thrift.TApplicationException.read(TApplicationException.java:111)

at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:71)

at org.apache.hive.service.cli.thrift.TCLIService$Client.recv\_OpenSession(TCLIService.java:156)

at org.apache.hive.service.cli.thrift.TCLIService$Client.OpenSession(TCLIService.java:143)

at org.apache.hive.jdbc.HiveConnection.openSession(HiveConnection.java:583)

at org.apache.hive.jdbc.HiveConnection.<init>(HiveConnection.java:192)

at org.apache.hive.jdbc.HiveDriver.connect(HiveDriver.java:105)

at java.sql.DriverManager.getConnection(DriverManager.java:664)

at java.sql.DriverManager.getConnection(DriverManager.java:247)

at com.kit.JDBCUtils.getConnection(JDBCUtils.java:50)

at com.kit.KfkToHiveFilter.getRULE\_RANGE\_COL\_NAME\_Set(KfkToHiveFilter.java:915)

at com.kit.KfkToHiveFilter.checkData(KfkToHiveFilter.java:217)

at com.kit.KfkToHiveFilter.access$2(KfkToHiveFilter.java:148)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:135)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:1)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at scala.collection.Iterator$class.foreach(Iterator.scala:727)

at org.apache.spark.util.NextIterator.foreach(NextIterator.scala:21)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:66)

at org.apache.spark.scheduler.Task.run(Task.scala:88)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:214)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)

at java.lang.Thread.run(Thread.java:745)

java.sql.SQLException: Could not establish connection to jdbc:hive2://172.16.19.154:10000/default: Required field 'client\_protocol' is unset! Struct:TOpenSessionReq(client\_protocol:null, configuration:{use:database=default})

at org.apache.hive.jdbc.HiveConnection.openSession(HiveConnection.java:594)

at org.apache.hive.jdbc.HiveConnection.<init>(HiveConnection.java:192)

at org.apache.hive.jdbc.HiveDriver.connect(HiveDriver.java:105)

at java.sql.DriverManager.getConnection(DriverManager.java:664)

at java.sql.DriverManager.getConnection(DriverManager.java:247)

at com.kit.JDBCUtils.getConnection(JDBCUtils.java:50)

at com.kit.KfkToHiveFilter.getRULE\_RANGE\_COL\_NAME\_Set(KfkToHiveFilter.java:915)

at com.kit.KfkToHiveFilter.checkData(KfkToHiveFilter.java:217)

at com.kit.KfkToHiveFilter.access$2(KfkToHiveFilter.java:148)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:135)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:1)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at scala.collection.Iterator$class.foreach(Iterator.scala:727)

at org.apache.spark.util.NextIterator.foreach(NextIterator.scala:21)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:66)

at org.apache.spark.scheduler.Task.run(Task.scala:88)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:214)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)

at java.lang.Thread.run(Thread.java:745)

Caused by: org.apache.thrift.TApplicationException: Required field 'client\_protocol' is unset! Struct:TOpenSessionReq(client\_protocol:null, configuration:{use:database=default})

at org.apache.thrift.TApplicationException.read(TApplicationException.java:111)

at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:71)

at org.apache.hive.service.cli.thrift.TCLIService$Client.recv\_OpenSession(TCLIService.java:156)

at org.apache.hive.service.cli.thrift.TCLIService$Client.OpenSession(TCLIService.java:143)

at org.apache.hive.jdbc.HiveConnection.openSession(HiveConnection.java:583)

... 24 more

org.apache.spark.Logging$class.logError(Logging.scala:96):10:07:42

ERROR:Exception in task 5.0 in stage 317.0 (TID 2858)

java.lang.NullPointerException

at com.kit.KfkToHiveFilter.getRULE\_RANGE\_COL\_NAME\_Set(KfkToHiveFilter.java:917)

at com.kit.KfkToHiveFilter.checkData(KfkToHiveFilter.java:217)

at com.kit.KfkToHiveFilter.access$2(KfkToHiveFilter.java:148)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:135)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:1)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at scala.collection.Iterator$class.foreach(Iterator.scala:727)

at org.apache.spark.util.NextIterator.foreach(NextIterator.scala:21)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:66)

at org.apache.spark.scheduler.Task.run(Task.scala:88)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:214)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)

at java.lang.Thread.run(Thread.java:745)

org.apache.spark.Logging$class.logWarning(Logging.scala:71):10:07:42

WARN:Lost task 5.0 in stage 317.0 (TID 2858, localhost): java.lang.NullPointerException

at com.kit.KfkToHiveFilter.getRULE\_RANGE\_COL\_NAME\_Set(KfkToHiveFilter.java:917)

at com.kit.KfkToHiveFilter.checkData(KfkToHiveFilter.java:217)

at com.kit.KfkToHiveFilter.access$2(KfkToHiveFilter.java:148)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:135)

at com.kit.KfkToHiveFilter$1$1.call(KfkToHiveFilter.java:1)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at org.apache.spark.api.java.JavaRDDLike$$anonfun$foreach$1.apply(JavaRDDLike.scala:330)

at scala.collection.Iterator$class.foreach(Iterator.scala:727)

at org.apache.spark.util.NextIterator.foreach(NextIterator.scala:21)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.rdd.RDD$$anonfun$foreach$1$$anonfun$apply$28.apply(RDD.scala:890)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.SparkContext$$anonfun$runJob$5.apply(SparkContext.scala:1850)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:66)

at org.apache.spark.scheduler.Task.run(Task.scala:88)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:214)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)

at java.lang.Thread.run(Thread.java:745)

org.apache.spark.Logging$class.logError(Logging.scala:75):10:07:42

G\_DM\_FUNCTION\_LOCATION

备份：

create table G\_DM\_FUNCTION\_LOCATION\_bak as select \* from G\_DM\_FUNCTION\_LOCATION ;

commit;

insert into G\_DM\_FUNCTION\_LOCATION

(ID,CLASSIFY\_ID,FL\_NAME,FL\_TYPE,RUNNING\_STATE,BASE\_VOLTAGE\_ID,OWNER\_STATION,SITE\_ID,SITE\_NAME,FEEDER\_ID,ASSETS\_PROPERTY,LINE\_FUNCTION,AREA\_FEATURE,XFMR\_TYPE,LATITUDE,LONGITUDE,POWER\_GRID\_FLAG,IS\_DEDICATEDLINE,DISPATCH\_NO,RUN\_DATE,POLE\_ID,SRC\_ID,OH\_LENGTH,CABLE\_LENGTH,LINE\_CATEGORY\_CODE,USER\_CODE,PARENT\_ID,MAIN\_LINE,P\_CODE,C\_CODE,B\_CODE,D\_CODE,FULL\_PATH)

select to\_number('20170215003'||2),CLASSIFY\_ID,FL\_NAME,FL\_TYPE,RUNNING\_STATE,BASE\_VOLTAGE\_ID,OWNER\_STATION,SITE\_ID,SITE\_NAME,FEEDER\_ID,ASSETS\_PROPERTY,LINE\_FUNCTION,AREA\_FEATURE,XFMR\_TYPE,LATITUDE,LONGITUDE,POWER\_GRID\_FLAG,IS\_DEDICATEDLINE,DISPATCH\_NO,RUN\_DATE,POLE\_ID,SRC\_ID,OH\_LENGTH,CABLE\_LENGTH,LINE\_CATEGORY\_CODE,USER\_CODE,PARENT\_ID,MAIN\_LINE,P\_CODE,C\_CODE,B\_CODE,D\_CODE,FULL\_PATH

from G\_DM\_FUNCTION\_LOCATION t where rownum = 1;

commit;

错误异常：

java.lang.RuntimeException: [2.10] failure: ``)'' expected but 1 found

(select 1

^