Migrate REDI Sybase stored procedures to Apache Phoenix

Contents

Document Revision History

Version	Author	Section Name	Description of Changes
1.0	Gabi Florea		Initial version
	Hariprasad Allaka	Implement RANK function in Phoenix	Added RANK solution

Replace Sybase UPDATE commands

Sybase:

UPDATE RAW_RS_TRANS_MASTER

Becomes UPSERT in Phoenix:

***UPSERT INTO** RAW_RS_TRANS_MASTER(Client12)*

Replace Sybase SET commands

Sybase: SET Client12=ClientId || SubClientId;

Becomes UPSERT with SELECT in Phoenix:

SELECT ClientId || SubClientId **FROM** MODS.RAW_RS_TRANS_MASTER;

Replace Sybase compare with empty characters

Sybase: WHERE coalesce(ItemCarrier,'')<>''

Becomes in Phoenix: **WHERE** ItemCarrier <> ’’

Replace Sybase string functions

LEFT

Sybase: GraphLabelWeek = *"left"*(WeekTextShort,5),

Becomes in Phoenix: *substr*(WeekTextShort,0,5)

RIGHT

Sybase: ""right""(CardNoMask,4)

Becomes in Phoenix: *SUBSTR(CardNoMask, -4)*

REPLACE

Sybase: REPLACE(TransactionId,'U','US')

Becomes in Phoenix: REGEXP_REPLACE(TransactionId,'U','US')

LOCATE

Sybase: set RawDataFileDate = substr(SourceFile,**locate**(SourceFile,'2013'),10)
Becomes in Phoenix: TO_DATE(SUBSTR(SourceFile,**INSTR**(SourceFile,'2013'),10), 'yyyy-MM-dd')

Replace Sybase DATE functions

Sybase: dateformat(dateadd(dd,-1,now()),'' yyyy-mm-dd'')
Becomes in Phoenix: TO_DATE(TO_CHAR((now() - 1), 'yyyy-MM-dd'))

OBS: Do not use: ***TO_DATE(TO_CHAR(NOW()),'YYYY-MM-DD')***
DD instead of dd, will return bad date.

Replace Sybase math functions

MOD

Sysbase: mod(ChargebackId,10000000)
Becomes in Phoenix: *ChargebackId - ROUND(ChargebackId/10000000) * 10000000*

Replace Sybase DEFAULT constraints

Sybase: CREATE TABLE MODS.WRK_CB2_TRANS_MASTER
(ChargebackYN char(1) DEFAULT 'N')

Becomes in Phoenix:
UPSERT INTO MODS.WRK_CB2_TRANS_MASTER(OID,ChargebackYN)
SELECT
OID, coalesce(ChargebackYN,'N') **FROM** mods.wrk_cb2_trans_master;

Replace Sybase TRUNCATE DDL command

Sybase: TRUNCATE TABLE mods.wrk_cb2_trans_master;

Becomes in Phoenix: **DELETE FROM** mods.wrk_cb2_trans_master;

Replace Sybase INNER JOIN SELECTS

Sybase: INNER JOIN (SELECT … FROM … GROUP BY
    ClientId) B ON A.ClientId=B.ClientId

Becomes in Phoenix:

INNER JOIN (**SELECT** … **FROM** … **GROUP BY** ClientId) B **ON** A.ClientId=B.ClientId

Replace Sybase temporary tables

We identified two types of uses of the temporary tables in Sybase:

- A. without any modification of the temporary data:

Here are some sample code:

```
Sybase:  SELECT OID INTO #tmp FROM MODS.RAW_RS_TRANS_DETAIL WHERE
    ucase(ProdDesc) LIKE '%GIFT%CARD%';
    UPDATE MODS.RAW_RS_TRANS_MASTER as A
        SET A.FLAG20 = 'G' FROM
    MODS.RAW_RS_TRANS_MASTER as A JOIN #tmp
    as B ON A.OID = B.OID WHERE COALESCE(A.FLAG20,'N') = 'N’;
```

Becomes in Phoenix:

```
    UPSERT INTO mods.raw_rs_trans_master(Flag20,OID)
    SELECT 'G', c.OID
    FROM
    (SELECT OID FROM mods.raw_rs_trans_master as a WHERE
    COALESCE(a.FLAG20,'N') = 'N') as c
    JOIN
    (SELECT OID FROM mods.raw_rs_trans_detail as b WHERE
    (upper(b.ProdDesc) like '%GIFT%CARD%')) as d ON c.OID = d.OID);
```

```
Sybase: select MAX(ChargebackId) as xBase into #tmpKey from MODS.CB_MASTER_DATA
 where ChargebackId < 40000000;
      update MODS.RAW_CB_MASTER_DATA
       set ChargebackId = ChargebackId+xBase from  MODS.RAW_CB_MASTER_DATA,#tmpKey;
      drop table #tmpKey;
```

Becomes in Phoenix:

```
UPSERT INTO MODS.RAW_CB_MASTER_DATA(ChargebackId)
SELECT A.ChargebackId + B.XBase FROM
MODS.CB_MASTER_DATA AS A, (SELECT MAX(ChargebackId) AS XBase FROM
MODS.CB_MASTER_DATA WHERE ChargebackId < 40000000) AS B;
```

- B. with modification of the temporary data:

```
Sybase: select * into #tmpDemo from MODS.RAW_CB_MASTER_DATA
       where ClientId = '000050'
       and(TransactionId like '%10' or TransactionId like '%59' or TransactionId like '%19' or TransactionId like
     '%23');
      update #tmpDemo
       set ChargebackId = ChargebackId-4000000,
       ClientId = 'DEMOJD',
       SubClientId = '000001',
       TChargebackId = trim(cast(mod(ChargebackId,10000000)-4000000 as varchar(10)));
      insert into MODS.RAW_CB_MASTER_DATA
       select * from #tmpdemo;
      drop table #tmpdemo;
```

Becomes in Phoenix:

```
UPSERT INTO MODS.RAW_CB_MASTER_DATA(ChargebackId, ClientId, SubClientId, TChargebackId)
SELECT A.ChargebackId, A.ClientId, A.SubClientId, A.TChargebackId
FROM (SELECT ChargebackId-4000000 as ChargebackId, 'DEMOJD' AS ClientId, '000001' AS
SubClientId, trim(TO_CHAR((ChargebackId - ROUND(ChargebackId/10000000) * 10000000) -
4000000)) AS TChargebackId
FROM MODS.RAW_CB_MASTER_DATA
WHERE ClientId = '000152'
) AS A;
```

Replace Sybase autoincrement

Sybase: ProdLogSeq integer not null default autoincrement

Becomes in Phoenix:

- In the create table file the column will be declared as follows: **PRODLOGSEQ INTEGER NOT NULL ;**
- Create a sequence:

```
CREATE SEQUENCE IF NOT EXISTS MODS.SEQ_RBI_PROCESS_LOG;
```

- On insert use:

```
UPSERT INTO MODS.RBI_PROCESS_LOG (ProdLogSeq,ProdGroup,ProdJob,ProdTime,ProdDate,ProdLogType,ProdStatus,ProdText)
values ( NEXT VALUE FOR MODS.ProdLogSeq
,'BEDC','BED_CB_001_Prepare',TO_DATE(TO_CHAR(Now())),TO_DATE(TO_CHAR(Now())),'JobStart','Starting','Process Start') ;
```

Replace Sybase Non primary Key tables

Phoenix requires a primary key for every table, so for non PK tables, use ROW_TIMESTAMP as PK:

```
CREATE TABLE EMPLOYEE_TEST
(PK DATE NOT NULL,
NAME VARCHAR,
SALARY INTEGER,
DEPARTMENT INTEGER,
CONSTRAINT PK_EMPLOYEE_TEST PRIMARY KEY(PK ROW_TIMESTAMP);

UPSERT INTO EMPLOYEE_TEST(NAME, SALARY, DEPARTMENT)
VALUES ('E1', 1000, 101);
```

Obs. Notice – no need to explicitly insert into the PK column.

https://phoenix.apache.org/rowtimestamp.html

Load CSV data into HBASE tables

Ex:

```
/usr/hdp/2.4.0.0-169/phoenix/bin/psql.py localhost:2181:/hbase-unsecure -t MODS.BED_CB_DATA -h in-line /root/prabhu/migration/data_dump/BED_CB_DATA.csv
```

If error occurs – check the column header is the same as in DB (upper case).

Sequence to replace ROWID

```
---Call a Sequence MODS.SEQ_RAW_BED_CB_ID

UPSERT INTO  MODS.RAW_BED_CB_DATA (OID,ChargebackId)
SELECT
 OID
 ,(NEXT VALUE FOR MODS.SEQ_RAW_BED_CB_ID)+ A.LastId
 --,ROWID(RAW_BED_CB_DATA)+ A.LastId /* find out the method for ROWID */
 FROM
 (select cast(max(COALESCE(ChargebackId,0)) as bigint) as LastId  from MODS.BED_CB_DATA)A
 ,MODS.RAW_BED_CB_DATA;
```

4. While creating secondary index on HBASE table column qualifies make sure the property "hbase.regionserver.wal.codec property" set to "org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec"

Implement RANK function in Phoenix

Using PIG: (Preferred and Followed)

Assumption:

Before we proceed to the further content of this document it is assumed that all the required applications are installed on the VM or local machine to integrate HBASE, Phoenix and PIG.

Objective:

The main objective is to migrate the RANK () windowing function from Sybase IQ to PIG in the data workflow.

Pre-requisites:

1. HBASE master and Region server should be installed and up and running

2. PIG should be installed and up and running.

Solution:

1. Download datafu.jar file and place it **"../pig/lib"** directory. (DataFu is a collection of user-defined functions for working with large-scale data in Hadoop and Pig. This library was born out of the need for a stable, well-tested library of UDFs for data mining and statistics.)

2. Navigate to **"../phoenix/lib"** directory and copy all the phoenix related JAR files to **"../pig/lib"**

3. Some of the phoenix jars like *client.jar are placed in **"../phoenix"** and those JAR files also need to be copied to **"../pig/lib"**.

4. In the PIG Latin script

    a. REGISTER all the jar files which are present in the "../pig/lib/" directory.

```
REGISTER /usr/hdp/2.4.0.0-169/pig/lib/*.jar;
```

b. Define the required UDF to be used from Datafu

```
DEFINE Enumerate datafu.pig.bags.Enumerate('1');
```

c. Load the source data using below commands

```
data = LOAD 'hbase://query/select ClientId,ClientName from
RBI_REF_CLIENT_MENUS' USING
org.apache.phoenix.pig.PhoenixHBaseLoader('localhost') as (ClientId:charArray,
ClientName:charArray);
```

d. Implement the logic to achieve final result

e. Store the result back to source using below command

```
STORE data INTO 'hbase://REF_MENU_CLIENT_SORT' USING
org.apache.phoenix.pig.PhoenixHBaseStorage('localhost', '-batchSize 5000')
```

f. Sample code is placed in below SVN location

https://svn.aciworldwide.com/repos-aci/ReD_DataWarehouse/Hadoop_Migration/Converted/Phoenix/Procedures/REF_RecreateClientMenus

Notes:

1. **localhost** refers to the host on which **HBASE** is running. This can be replaced by Hostname on which **HBASE** is running, if the **PIG Latin** script is running in some other server.
2. All the paths mentioned in the solution are relative to the installation paths and are subjected to change based on the Hadoop distribution and its type of installation.

Other solutions:

1. The same can be implemented by integrating HBASE and Phoenix with
   a. SPARK
   b. HIVE
   c. JAVA MAPREDUCE
   d. APACHE DRILL

Using HIVE:

Problem Statement:

RANK() windowing function is not available in phoenix right now.

Possible Solutions:

Solution	Pros	Cons

Java	1. Single point of solution 2. Reusable code	1. Lacking in expertise 2. Time consuming with deadlines
Hive <-> HBASE <-> Phoenix	1. Have SQL based expertise to address it 2. Faster in terms of delivery	1. Hive can be on slower side in terms of processing 2. Needs some extra configuration for Hbase storage handler 3. Code repeating in all the places
Apache Drill	1. Have the querying capabilities on HBase tables 2. Rank is already implemented in Querying language	1. Needs to gain expertise 2. Time needed to find out how does this fit into OOZIE workflow 3. Need to find out read/write of Drill on Hbase

Solve it using Hive:

Out of 3 solutions mentioned above, we are trying implementing this using Hive (2nd in the table above).

The solutions can be explained in few steps mentioned below

1. Break the Phoenix procedure into multiple code blocks , to be pushed into Oozie work flow
2. The code blocks contains
    a. Phoenix code blocks

       Where ever we can have the code written in phoenix directly in the sequence of the procedure, we place the code in Phoenix by creating required tables in Phoenix.

    b. Hive Code blocks

       Whenever we come across RANK function implemented in existing SYBASE IQ procedures, we take that code and

        i. Create external tables in HIVE using HBASE storage handler for the tables getting referred in the existing Sybase code.

ii. Create an external table using HBASE storage handler to store the result set of query using RANK function.

iii. Push the data into table created in step ii, using Hive queries.

iv. Create the same structure as external table created in step ii in phoenix and HBASE and start using the table names in phoenix scripts for further processing.

Questions:

1. Will this implementation make the process slow?

   Ans: Yes, As HIVE is basically for batch processing and performs for read-only, the map reduce code built in the back ground while executing queries will result slowness in the process.

2. Can we tune this further in product road map?

   Ans: YES,There are tuning mechanisms using which we can tune this process. But we need to check the usability of the tuning process on our application.

3. Can we any time replace this code with minimum effort when RANK is available in phoenix?

   Ans: YES, According to Phoenix documentation, they have RANK in their roadmap and we can have it in our **code as part of a sprint once it's available.**

Important Configurations

Sno	Property name	Property value	Location	significance
1	hbase.regionserver.wal.codec property	org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec	Hbase-site.xml	Enables secondary indexing in HBASE