Recursive SQL for Sanity and Performance

Baltimore/Washington DB2 Users' Group
14 September 2016

Veasna Tauch
vtauch@atpco.net
Airline Tariff Publishing Company

Agenda

- Problem Statement
- Solution Approach & Alternatives
- Using Recursion & CTEs to Solve the Problem
- Practical Applications of Recursion & CTE
- Summary
- Questions

Problem Statement

- Dynamic SQL statements can get very long and perform poorly because of
  - Application frameworks generating SQLs
  - Tools such as Hibernate generating SQL
  - “One size fits all” approach
  - Pushing DB2 to do more in a single statement/request

Problem Statement

- What’s wrong with a long SQL?
  - Hard to debug and tune
  - Increases PREPARE/BIND time
  - Impacts Dynamic SQL Statement Cache
    - ✓ Maximum cache size < 1GB
    - ✓ SQL statement maximum length = 2MB

Problem Statement

Original SQL Design

This SQL contains 2,400+ sets of predicates and about 177 pages.

```
SELECT CARRIER_CD
      ,FILE_DT
      ,SEQ_NO

      ,…
  FROM FARE_SUBSCRIPT_TBL
 WHERE STAT_CD      = 'A'
   AND (CARRIER_CD = 'AA'
   AND  FILE_DT    = '01/01/2016'
   AND  SEQ_NO     = 1
    OR  CARRIER_CD = 'AA'
   AND  FILE_DT    = '01/01/2016'
   AND  SEQ_NO     = 2

… Repeated 2,400+ times
    OR  CARRIER_CD = '9W'
   AND  FILE_DT    = '01/05/2016'
   AND  SEQ_NO     = 999)
 ORDER BY …
```


ATPCO

Solution Approach & Alternatives

Would it be better if we could load those sets of parameters into a DB2 table, then INNER JOIN this parameter table with the targeted table?


```
FROM FARE_SUBSCRIPT_TBL

  AND (CARRIER_CD = 'AA'
  AND  FILE_DT    = '01/01/2016'
  AND  SEQ_NO     = 1
   OR  CARRIER_CD = 'AA'
  AND  FILE_DT    = '01/01/2016'
  AND  SEQ_NO     = 2
… Repeated 2,400+ times
   OR  CARRIER_CD = '9W'
  AND  FILE_DT    = '01/05/2016'
  AND  SEQ_NO     = 999)
```

INNER JOIN

CARRIER_CD	FILE_DT	FILE_SEQ_NO
AA	01/01/2016	1
AA	01/01/2016	2
…	…	…
9W	01/31/2016	999

Transform

ATPCO

Solution Approach & Alternatives

What kind of table should we use for the **transient** data?

- Regular DB2 Table
  - Maintenance: Clean up after use
  - Could create contention during INSERT and DELETE
- Global Temporary Table (GTT)
  - No cleanup and no I/O contention among threads
  - Require multiple requests to DB2: Create, Insert, Select and Drop
- Common Table Expression (CTE)
  - ✓ No cleanup and no I/O contention among threads
  - ✓ A **single** request to DB2

Solution Approach & Alternatives

Using Common Table Expression, we will

1. Have the application create a string of parameters and load into the 1st CTE
2. Create a 2nd CTE to determine the number of parameter sets
3. Use Recursive SQL to parse the string in the 1st CTE and store in a 3rd CTE
4. Then INNER JOIN the 3rd CTE with the targeted table

That sounds simple, and it is that simple, and

Simplicity is beauty!

Using Recursion & CTE to Solve the Problem

```
 FROM FARE_SUBSCRIPT_TBL

  AND (CARRIER_CD = 'AA'
  AND  FILE_DT    = '01/01/2016'
  AND  SEQ_NO     = 1
   OR  CARRIER_CD = 'AA'
  AND  FILE_DT    = '01/01/2016'
  AND  SEQ_NO     = 2

… Repeated 2,400+ times
   OR  CARRIER_CD = '9W'
  AND  FILE_DT    = '01/05/2016'
  AND  SEQ_NO     = 999)
```

INNER JOIN

CARRIER_CD	FILE_DT	FILE_SEQ_NO
AA	01/01/2016	1
AA	01/01/2016	2
…	…	…
9W	01/31/2016	999

Parse string using *Recursive SQL* and store the data in a CTE

```
AA01/01/2016001,AA01/01/2016002,…
9W01/05/2016999
```

Transformed by Application

Using Recursion & CTE to Solve the Problem

Syntax Overview

DETOUR

Objective: To calculate BWDB2UG Meeting date from 2017 through 2021.
Rule: The quarterly meeting date is on the 2nd Wednesday of the 3rd month.

```
WITH MEET_DATE_TBL                                  -- CTE name
     (MEET_DT)                                      -- Column list
  AS (SELECT DATE('03/01/2017')                     -- Set initial value of LAST_FILE_DT
       FROM SYSIBM.SYSDUMMY1
      UNION ALL
     SELECT MEET_DT + 3 MONTHS                       -- Increment loop counter
       FROM MEET_DATE_TBL                           -- Table name is the same as CTE name
      WHERE MEET_DT < '12/01/2021')                 -- Loop control

SELECT CASE WHEN DAYOFWEEK(MEET_DT) = 4             -- Select statement that creates
            THEN NEXT_DAY(MEET_DT,'WED')            --        result set or report
            ELSE NEXT_DAY(NEXT_DAY(MEET_DT,'WED'),'WED')
       END "MEET DATE"
  FROM MEET_DATE_TBL;
```

Results

```
03/08/2017
06/14/2017
09/13/2017
12/13/2017
03/14/2018
06/13/2018
09/12/2018
12/12/2018
03/13/2019
06/12/2019
09/11/2019
12/11/2019
03/11/2020
06/10/2020
09/09/2020
12/09/2020
03/10/2021
06/09/2021
09/08/2021
12/08/2021
```

Using Recursion & CTE to Solve the Problem

```
FROM FARE_SUBSCRIPT_TBL

 AND (CARRIER_CD = 'AA'
 AND  FILE_DT    = '01/01/2016'
 AND  SEQ_NO     = 1
  OR  CARRIER_CD = 'AA'
 AND  FILE_DT    = '01/01/2016'
 AND  SEQ_NO     = 2

… Repeated 2,400+ times
  OR  CARRIER_CD = '9W'
 AND  FILE_DT    = '01/05/2016'
 AND  SEQ_NO     = 999)
```

INNER JOIN

CARRIER_CD	FILE_DT	FILE_SEQ_NO
AA	01/01/2016	1
AA	01/01/2016	2
…	…	…
9W	01/31/2016	999

Parse string using Recursive SQL and store the data in a CTE

```
AA01/01/2016001,AA01/01/2016002,…
9W01/05/2016999
```

Transformed by Application

Using Recursion & CTE to Solve the Problem

```
AA01/01/2016001,AA01/01/2016002,…
9W01/05/2016999
```

1

2nd CTE to determine the number of parameter sets

2

CARRIER_CD	FILE_DT	FILE_SEQ_NO
AA	01/01/2016	1
AA	01/01/2016	2
…	…	…
9W	01/31/2016	999

3

INNER JOIN the 3rd CTE with the targeted table

4

```
WITH IP_PARM_ARRAY (IP_PARM_STR)
  AS (SELECT 'AA01/01/2016001,AA01/01/2016002,AA01/01/2016011,' ||
             'AB01/01/2016003,AC01/01/2016002,AC01/01/2016099,' ||
             …
             'YY01/01/2016099,YY01/01/2016123,1A01/05/2016888,' ||
             '9W01/05/2016999'
        FROM SYSIBM.SYSDUMMY1)

   ,IP_PARM_COUNT (IP_PARM_NO)
  AS (SELECT INT((LENGTH(IP_PARM_STR) + 1) / POSSTR(IP_PARM_STR,','))
        FROM IP_PARM_ARRAY)

,IP_PARM_RSET (LOOP_CT, STR_POS, PK_COL1, PK_COL2, PK_COL3)
  AS (SELECT 1
            ,16
            ,CHAR(SUBSTR(A.IP_PARM_STR, 1, 2),2)
            ,DATE(SUBSTR(A.IP_PARM_STR, 3,10))
            ,INT (SUBSTR(A.IP_PARM_STR,13, 3))
       FROM IP_PARM_ARRAY A
      UNION ALL
      SELECT C.LOOP_CT + 1
            ,C.STR_POS + 16
            ,CHAR(SUBSTR(A.IP_PARM_STR, C.STR_POS +  1, 2),2)
            ,DATE(SUBSTR(A.IP_PARM_STR, C.STR_POS +  3,10))
            ,INT (SUBSTR(A.IP_PARM_STR, C.STR_POS + 13, 3))
        FROM IP_PARM_ARRAY A
            ,IP_PARM_COUNT B
            ,IP_PARM_RSET  C
       WHERE C.LOOP_CT < B.IP_PARM_NO)

SELECT B.CARRIER_CD
      ,B.FILE_DT
      ,B.SEQ_NO
      ,…
  FROM IP_PARM_RSET      A INNER
  JOIN FARE_SUBSCRIPT_TBL B
    ON A.PK_COL1 = B.CARRIER_CD
   AND A.PK_COL2 = B.FILE_DT
   AND A.PK_COL3 = B.SEQ_NO
 WHERE B.STAT_CD  = 'A'
 ORDER BY B.CARRIER_CD, B.FILE_DT, B.SEQ_NO;
```

Step 1. Application Create String Parameters

```
/*Load input parameters into a CTE.  Use fixed length and comma delimited for simplicity & readability */

WITH IP_PARM_ARRAY
     (IP_PARM_STR)
  AS (SELECT 'AA01/01/2016001,AA01/01/2016002,AA01/01/2016011,' ||
             'AB01/01/2016003,AC01/01/2016002,AC01/01/2016099,' ||
             …
             'YY01/01/2016099,YY01/01/2016123,1A01/05/2016888,' ||
             '9W01/05/2016999'
        FROM SYSIBM.SYSDUMMY1)
```

Notes:
1. This CTE will have a column and a row.
2. As of DB2 V10, SELECT ? Is not permitted in , but SELECT CAST(? AS VARCHAR(30000)) is permitted.
3. CTE's row length is limited to 32K. If the parameter string is longer than 32K, we will have to create multiples of this CTE.

Step 2. CTE to Determine the Number of Parameters

```
/*  Determine number of parameter sets, which will be used as a loop control later.
    Number of Parameter Set = (Length of Input String + 1) / Position of first comma   */

    ,IP_PARM_COUNT
     (IP_PARM_NO)
  AS (SELECT INT((LENGTH(IP_PARM_STR) + 1)
                 / POSSTR(IP_PARM_STR, ','))
        FROM IP_PARM_ARRAY)
```

Notes:
1. After execution, this CTE will have one column and one row.
2. The application program could determine the value up front and eliminate the need to have this CTE.

Step 3. Recursive SQL Parses the String

/* Parse the IP_PARM_ARRAY.IP_PARM_STR into 3 columns. The first 2 columns are for housekeeping. */

```
    ,IP_PARM_RSET
    (LOOP_CT, STR_POS, PK_COL1, PK_COL2, PK_COL3)
AS (SELECT 1                                          -- This portion of SELECT statement generates the
          ,16                                         --      1st row and executed once.
          ,CHAR(SUBSTR(A.IP_PARM_STR, 1, 2), 2)
          ,DATE(SUBSTR(A.IP_PARM_STR, 3,10))          -- These substring values are converted to match
          ,INT (SUBSTR(A.IP_PARM_STR,13, 3))          --      target table's columns data type.
      FROM IP_PARM_ARRAY A
     UNION ALL                                        -- This portion of SELECT is executed 2nd time forward
    SELECT C.LOOP_CT + 1                              -- Increment the loop counter
          ,C.STR_POS + 16                             -- Increment string offset
          ,CHAR(SUBSTR(A.IP_PARM_STR, C.STR_POS +  1, 2), 2)
          ,DATE(SUBSTR(A.IP_PARM_STR, C.STR_POS +  3,10))
          ,INT (SUBSTR(A.IP_PARM_STR, C.STR_POS + 13, 3))
      FROM IP_PARM_ARRAY A                            -- First 2 input CTEs (A & B) have 1 row each. So, it is
          ,IP_PARM_COUNT B                            --      safe to do a Cartesian product (implicit
          ,IP_PARM_RSET  C                            --      INNER JOIN without join predicates).
     WHERE C.LOOP_CT < B.IP_PARM_NO)
```

Step 4. INNER JOIN the CTE with the Target Table

```
SELECT B.CARRIER_CD
      ,B.FILE_DT
      ,B.SEQ_NO
      ,…
  FROM IP_PARM_RSET       A INNER
  JOIN FARE_SUBSCRIPT_TBL B
    ON A.PK_COL1 = B.CARRIER_CD
   AND A.PK_COL2 = B.FILE_DT
   AND A.PK_COL3 = B.SEQ_NO
 WHERE B.STAT_CD = 'A'
 ORDER BY B.CARRIER_CD
         ,B.FILE_DT
         ,B.SEQ_NO;
```

Note: This concept would work well only when those columns are index columns, or leading index columns.

Practical Applications of Recursion & CTE

1. Preload reference data
2. Select data *not in the database*
3. Generate Date Dimension Table for data warehouse

ATPCO

Preload Reference Data

Objective: Prepopulate a table with 2 columns – A DATE and CHAR(3). The data is inserted directly into a table using an INSERT statement.

```
/* Data is being generated by a CTE and loaded into a table in a single step */

INSERT INTO LAST_FILE_INFO
       (LAST_FILE_DT
       ,LAST_FILE_CD)
WITH LAST_FILE_TBL
     (LAST_FILE_DT)
  AS (SELECT CURRENT DATE
        FROM SYSIBM.SYSDUMMY1
       UNION ALL
      SELECT LAST_FILE_DT + 1 DAYS
        FROM LAST_FILE_TBL
       WHERE LAST_FILE_DT < '12/31/2025')

SELECT LAST_FILE_DT
      ,'AA '
  FROM LAST_FILE_TBL;
```

Select Data Not in the Database

Objective: Determine RULE codes, which are not being used for a given Carrier, Tariff, and RULE range.

```
WITH AVAIL_RULE_LIST
     (CARRIER, TARIFF, RULE)
  AS (SELECT 'AA', 'XYZ1234'
            ,'AB00'                  -- From RULE
        FROM SYSIBM.SYSDUMMY1
       UNION ALL
      SELECT 'AA', 'XYZ1234'
            ,ATP.NEXT_RULE(RULE)     -- Increment RULE UDF()
        FROM AVAIL_RULE_LIST
       WHERE RULE < 'AB99')          -- To RULE

SELECT A.RULE
  FROM AVAIL_RULE_LIST   A LEFT      -- Anti-JOIN
  JOIN RULE_CATEGORY_TBL B
    ON A.CARRIER  = B.CARRIER
   AND A.TARIFF   = B.TARIFF
   AND A.RULE     = B.RULE
 WHERE B.CARRIER IS NULL;            -- Anti-JOIN
```

Generate Date Dimension Table

Objective: Create Date Dimension reference table that will be used for data warehouse. Precompute date attributes: Day of Week, Day of Year, Week of Year, Month Name, Quarter Number, and Federal Holiday Name.

Federal Holidays

1.	New Year's Day	Fixed on 1 January
2.	Birthday of Martin Luther King, Jr.	3rd Monday in January
3.	President Day	3rd Monday in February
4.	Memorial Day	Last Monday in May
5.	Independence Day	Fixed on 4 July
6.	Labor Day	1st Monday in September
7.	Columbus Day	2nd Monday in October
8.	Veterans Day	Fixed on 11 November
9.	Thanksgiving Day	Fourth Thursday in November
10.	Christmas Day	Fixed on 25 December

Generate Date Dimension Table

Will have four CTEs, which will

1. Create an array of dates from '2015-01-01' through '2025-12-31'
2. Create an array of 2 columns and 12 rows of Month Number and Name
3. Create estimate dates for Federal holidays from '2014-01-01' through '2025-12-31'
4. Read back the 3rd CTE and adjust Federal holidays based on rules

Final SELECT statement will generate data for Date Dimension table

ATPCO

Generate Date Dimension Table

1

```
WITH CTE_ALL_DATE
     (CALENDAR_DT)
  AS (SELECT DATE('01/01/2015')
        FROM SYSIBM.SYSDUMMY1
       UNION ALL
      SELECT CALENDAR_DT + 1 DAYS
        FROM CTE_ALL_DATE
       WHERE CALENDAR_DT < '12/31/2025')
```

2

```
    , CTE_MONTH_NAME
     (MONTH_NO, MONTH_TXT)
  AS (SELECT  1, 'January  ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  2, 'February ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  3, 'March    ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  4, 'April    ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  5, 'May      ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  6, 'June     ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  7, 'July     ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  8, 'August   ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT  9, 'September' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT 10, 'October  ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT 11, 'November ' FROM SYSIBM.SYSDUMMY1 UNION ALL
      SELECT 12, 'December ' FROM SYSIBM.SYSDUMMY1)
```

Generate Date Dimension Table

3a

```
 ,CTE_EST_FED_HOLIDAY
  (NEW_YEAR_DAY      -- Jan 1st
  ,MLK_BIRTHDAY      -- 3rd Mon in Jan
  ,PRESIDENT_DAY     -- 3rd Mon in Feb
  ,MEMORIAL_DAY      -- Last Mon in May
  ,INDEPENDENT_DAY   -- Jul 4th
  ,LABOR_DAY         -- 1st Mon in Sep
  ,COLUMBUS_DAY      -- 2nd Mon in Oct
  ,VETERANS_DAY      -- Nov 11th
  ,THANKSGIVING_DAY -- 4th Thu in Nov
  ,CHRISTMAS_DAY)    -- Dec 25th
 AS (SELECT DATE('2014-01-01')
           ,DATE('2014-01-14')
           ,DATE('2014-02-14')
           ,DATE('2014-05-24')
           ,DATE('2014-07-04')
           ,DATE('2014-08-31')
```

3b

```
           ,DATE('2014-10-07')
           ,DATE('2014-11-11')
           ,DATE('2014-10-31')
           ,DATE('2014-12-25')
     UNION ALL
    SELECT NEW_YEAR_DAY      + 1 YEARS
           ,MLK_BIRTHDAY     + 1 YEARS
           ,PRESIDENT_DAY    + 1 YEARS
           ,MEMORIAL_DAY     + 1 YEARS
           ,INDEPENDENT_DAY  + 1 YEARS
           ,LABOR_DAY        + 1 YEARS
           ,COLUMBUS_DAY     + 1 YEARS
           ,VETERANS_DAY     + 1 YEARS
           ,THANKSGIVING_DAY + 1 YEARS
           ,CHRISTMAS_DAY    + 1 YEARS
      FROM CTE_EST_FED_HOLIDAY
     WHERE NEW_YEAR_DAY < '2025-12-31')
```

Generate Date Dimension Table

4a

```
 ,CTE_ADJUST_FED_HOLIDAYS
  (HOLIDAY_DT, HOLIDAY_NM)
AS (SELECT CASE DAYOFWEEK(NEW_YEAR_DAY)
                WHEN 1 THEN NEW_YEAR_DAY + 1 DAYS
                WHEN 7 THEN NEW_YEAR_DAY - 1 DAYS
                       ELSE NEW_YEAR_DAY
           END
          ,'New Year'
      FROM CTE_EST_FED_HOLIDAY     UNION
    SELECT NEXT_DAY(MLK_BIRTHDAY,'MON')
          ,'Birthday of Martin Luther King, Jr'
      FROM CTE_EST_FED_HOLIDAY     UNION
    SELECT NEXT_DAY(PRESIDENT_DAY,'MON')
          ,'President Birthday'
      FROM CTE_EST_FED_HOLIDAY     UNION
    SELECT NEXT_DAY(MEMORIAL_DAY, 'MON')
          ,'Memorial Day'
      FROM CTE_EST_FED_HOLIDAY     UNION
    SELECT CASE DAYOFWEEK(INDEPENDENT_DAY)
                WHEN 1 THEN INDEPENDENT_DAY + 1 DAYS
                WHEN 7 THEN INDEPENDENT_DAY - 1 DAYS
                       ELSE INDEPENDENT_DAY
           END
          ,'Independence Day'
      FROM CTE_EST_FED_HOLIDAY     UNION
```

4b

```
SELECT NEXT_DAY(LABOR_DAY, 'MON')
      ,'Labor Day'
  FROM CTE_EST_FED_HOLIDAY        UNION
SELECT NEXT_DAY(COLUMBUS_DAY, 'MON')
      ,'Columbus Day'
  FROM CTE_EST_FED_HOLIDAY        UNION
SELECT CASE DAYOFWEEK(VETERANS_DAY)
            WHEN 1 THEN VETERANS_DAY + 1 DAYS
            WHEN 7 THEN VETERANS_DAY - 1 DAYS
                   ELSE VETERANS_DAY
       END
      ,'Veterans Day'
  FROM CTE_EST_FED_HOLIDAY        UNION
SELECT NEXT_DAY(THANKSGIVING_DAY,'THU') + 21 DAYS
      ,'Thanksgiving'
  FROM CTE_EST_FED_HOLIDAY        UNION
SELECT CASE DAYOFWEEK(CHRISTMAS_DAY)
            WHEN 1 THEN CHRISTMAS_DAY + 1 DAYS
            WHEN 7 THEN CHRISTMAS_DAY - 1 DAYS
                   ELSE CHRISTMAS_DAY
       END
      ,'Christmas'
  FROM CTE_EST_FED_HOLIDAY)
```

Generate Date Dimension Table

5

```
SELECT A.CALENDAR_DT             "Date"
      ,CASE DAYOFWEEK(A.CALENDAR_DT)
            WHEN 1 THEN 'Sun'
            WHEN 2 THEN 'Mon'
            WHEN 3 THEN 'Tue'
            WHEN 4 THEN 'Wed'
            WHEN 5 THEN 'Thu'
            WHEN 6 THEN 'Fri'
            WHEN 7 THEN 'Sat'
       END                         "Day of Week"
      ,DAYOFYEAR(A.CALENDAR_DT)    "Day of Year"
      ,WEEK(A.CALENDAR_DT)         "Week of Year"
      ,B.MONTH_TXT                 "Month Name"
      ,QUARTER(A.CALENDAR_DT)      "Quarter"
      ,HOLIDAY_NM                  "Holidays Name"
  FROM CTE_ALL_DATE   A INNER
  JOIN CTE_MONTH_NAME B
    ON MONTH(A.CALENDAR_DT)  = B.MONTH_NO LEFT
  JOIN CTE_ADJUST_FED_HOLIDAYS C
    ON A.CALENDAR_DT         = C.HOLIDAY_DT
 ORDER BY A.CALENDAR_DT;
```

Summary

Recursive SQL gives us a means to build a transient table, which can be used to

- ✓ Simplify and streamline the application process
- ✓ Improve SQL performance
- ✓ Eliminate a need for developing an ad hoc program, or a simple stored procedure
- ✓ Find data that is **not** in a database