Modernizing CA ADS™ Applications

Kurt McKinney
MPO

CA IDMS™ Technical Conference

Framingham MA
December 2-5, 2014

Abstract

- At our shop, CA ADS modules are taking on a new role as callable business logic in an SQL procedure. In this session we cover the new architecture for driving and running SQL procedures and how we've reorganized CA ADS code for this usage.

Learning objectives:

- How can the CA ADS business logic be reused in SQL procedures
- How can we separate CA ADS code into presentation and business logic

Biography

- Mr. McKinney has decades of CA IDMS programming experience in batch, online, COBOL, and CA ADS. When forced, PL/I.

Agenda

- Process XML transactions
- XML processing flow
- Overview of the CA ADS conversion logic
- SQL Procedure Creation Preparation
- SQL Procedure Creation
- SQL Procedure Execution
- SQL Procedure Considerations
- Future Direction

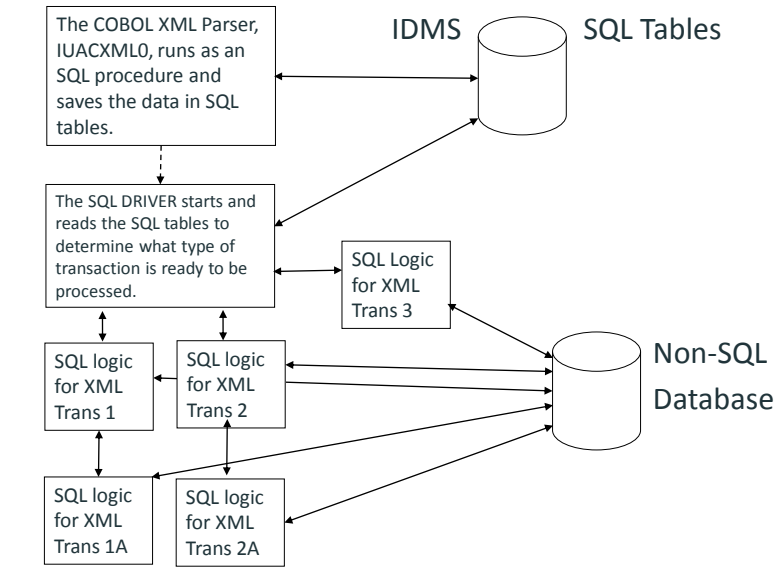
Process XML transactions

Processing XML transactions

- The new XML transactions are machine readable versions of the existing 3270 transactions
- One XML transaction does NOT translate to a single CA ADS Dialog
- The CA ADS code is already written (business logic)
- We only want one copy of the business logic
- Looking past the XML processing
  - Web pages that provide a better view of the data

XML processing flow

XML processing flow



Overview of the CA ADS conversion logic

Overview of the CA ADS conversion logic

- Goal: Minimize the changes to the code that works
- Begin with an CA ADS Dialog that performs the required business logic using a map and work records
- Divide the existing process code into include modules representing retrieval, validation, and update logic.
  - Look at the dialog structure, not the structure of the business logic
  - The dialog structure is more important than the business logic
  - If you understand how the dialogs are written, you shouldn't need to understand the business logic
- The Process code is split into INCLUDE modules that can be shared between the online and SQL procedure
- Repackage to run as a premap process

Overview of the CA ADS conversion logic

- SQL procedure Execution
  - The new SQL procedure code populates the existing map /work records
  - The SQL procedure code runs as close to original online as possible.
  - The SQL procedure pulls the results from the existing map /work records
- The result is two CA ADS Dialogs – an online version and the SQL version
- Try to minimize the need to be environment aware
  - Processing online or as an SQL procedure
  - Processing XML or driving a web site

SQL Procedure Creation Preparation

SQL Procedure Creation Preparation
'Rearrange' the existing CA ADS dialog

- First time, create a record that will be shared between the existing CA ADS dialog and the new SQL procedure
  - At a minimum this record would have RETURN-CODE and RETURN-MSG
  - Other Indicators: Transaction sharing, CA ADS or SQL, retrieval only
  - Work space for dynamically built SQL commands
- Add this record to the list of records associated with the existing dialog
  - This is a 'global' record and will be used in any subsequent iterations of this process
- Change the existing CA ADS dialog so it only has 1 exit point (display)
  - Change DISPLAYS that use literals to MOVE the text to RETURN-MSG
  - Change the DISPLAY to reference RETURN-MSG
  - Create a DISPLAY subroutine in an includable module
  - Call the subroutine where the DISPLAYS were coded

SQL Procedure Creation Preparation
'Rearrange' the existing CA ADS dialog

- Identify the block(s) of code in the existing CA ADS dialog that does all the data *retrieval* and populates the map
  - Generally this would be the premap but could be in a response process. Put this into an includable module
  - Understand the DIALOG structure. Not the business logic.
- Identify the block(s) of code in the existing CA ADS dialog that does the *validating* and *updating* of data
  - This is your 'business logic'
  - Again, put this code into an includable module (2 or more modules, if the validating and updating are done separately)

SQL Procedure Creation Preparation
'Rearrange' the existing CA ADS dialog

- Create or update Process modules, premap and response, to reference the new INCLUDE business logic components
  - The new premap process may only contain multiple INCLUDE statements
  - Use a naming convention to easily identify shared process code
- Change the existing CA ADS dialog to use the new includable modules
- Test the dialog to verify it functions EXACTLY this same

SQL Procedure Creation

SQL Procedure Creation
Build the SQL procedure

- In OCF, create a new SQL procedure
  - Associate it with a new CA ADS dialog

```
CREATE PROCEDURE DEMO.UPDATE_CONTACT  (
   CONTACT_NAME_IN                  CHARACTER(40),
   CONTACT_ADDRES_IN                CHARACTER(30),
   CONTACT_CITY_IN                  CHARACTER(20),
   CONTACT_STATE_IN                 CHARACTER(15),
   CONTACT_ZIP_IN                   CHARACTER(6),
   CONTACT_PHONE_IN                 CHARACTER(10),
   CONTACT_EMAIL_IN                 CHARACTER(25),
   RETURN_CODE_OUT                  DECIMAL(4),
   RETURN_MSG_OUT                   CHARACTER(80),
   MAP_RECORD_OUT                   CHARACTER(500) )
   EXTERNAL NAME DEMODIAG           LANGUAGE ADS
   PROTOCAL ADS                     DEFAULT DATABASE NULL
   ESTIMATED ROWS 1                 SYSTEM MODE
   TRANSACTION SHARING OFF          LOCAL WORK AREA 1024;
```

SQL Procedure Creation
Build the CA ADS SQL procedure logic

- Create a premap process for the new dialog
  - The premap needs to be able to perform the retrieval, validation, and update logic
  - Verify any required parameters from the SQL procedure are present and valid.
  - Check the validity of any optional parameters
- Prepare to retrieve business logic data
  - Move key values, if any, to the map record
  - Using the includable modules, perform data retrieval to populate the map record(s)

SQL Procedure Creation
Build the CA ADS SQL procedure logic

- Prepare to update using the business logic
  - If the retrieval was successful,
    - Move the field values that are to be updated to the map record
    - Using the includable modules, perform data *validation* and *updates*
- Exit the SQL procedure
  - Populate anything we want returned back (RETURN-CODE and RETURN-MSG at least)
  - We need a different includable module as an exit point for this dialog
  - The SQL procedure can't do a DISPLAY
  - It has to do a LEAVE ADS

SQL Procedure Creation
Build the CA ADS SQL procedure, ADSC

- Create the new dialog
  - Copy the already existing dialog
- Populate the access module field on the database specifications panel with the dialog name

```
                    Database Specifications
                      Dialog  DEMODIAG  Version     1
Subschema . . . . . . . . . . . .   DEMOSUBS
Schema  . . . . . . . . . . . . .   DEMOSCHM
Version . . . . . . . . . . . . .   0001

Access Module . . . . . . . . . .   DEMODIAG
SQL Compliance  . . . . . . . . .   _ 1. ANSI-standard SQL
                                      2. FIPS
Date Default Format . . . . . . .   _ 1. ISO 2. USA 3. EUR 4. JIS
Time Default Format . . . . . . .   _ 1. ISO 2. USA 3. EUR 4. JIS

   Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```


SQL Procedure Creation
Build the CA ADS SQL procedure, ADSC

- Add the SQL procedure definition to the list of records associated with the dialog

```
                Records and Tables          Page 1 of 1
             Dialog  DEMODIAG  Version     1

 Name                          Version  Work  New copy  Drop
1.DEMO.UPDATE_CONTACT________  ____     /       _        _
2._____________________________ ____     _       _        _
3._____________________________ ____     _       _        _

…

Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

SQL Procedure Creation
Build the CA ADS SQL procedure, ADSC

- Delete all the response processes and add the new single process (as premap)
- The MAP reference does remain to automatically include required records and control blocks
  - We did not remove MODIFY MAP commands
- You should now be able to impress your management by saying ‘look what I can do’

SQL Procedure Execution

SQL Procedure Execution

The new procedure is executed via a SQL select statement.

To change a contacts phone number –

```
EXEC SQL
     SELECT
          RETURN_CODE_OUT
        , RETURN_MSG_OUT
        INTO
          :RETURN-CODE
        , :RETURN-MSG
        FROM  DEMO.UPDATE_CONTACT
        (
           CONTACT_NAME_IN  = 'JOE SCHMOE'
           CONTACT_PHONE_IN = '2025550101'
        )
   END-EXEC.
```


SQL Procedure Execution

The new procedure is executed via a SQL select statement.

To dump the map record possibly to populate a web page–

```
EXEC SQL
     SELECT
          RETURN_CODE_OUT
        , RETURN_MSG_OUT
        , MAP_RECORD_OUT
        INTO
          :RETURN-CODE
        , :RETURN-MSG
        , :MAP-RECORD

    FROM  DEMO.UPDATE_CONTACT
        (
           CONTACT_NAME_IN  = 'JOE SCHMOE'
        )
   END-EXEC.
```

SQL Procedure Considerations

SQL Procedure Considerations

- Validation and editing by the map
  - Numeric fields
  - Edit and encode/decode tables
  - User written edit modules
  - The validation would have to be in the code for the SQL procedure. Yes, write some new code.
- Modify Map commands
  - No change required for field attributes (color, underscore, etc.)
- Conditional map commands (field changed, field in error)
  - Since there's no user interaction, these may/may not matter
  - Determine if they matter and deal with them accordingly

SQL Procedure Considerations

- Called programs and programs they call may also have to be reengineered
  - Are they doing a DISPLAY
  - Are they doing a ROLLBACK

SQL Procedure Considerations

- Transaction sharing
  - When enabled, it allows all SQL procedures to execute as a single recovery unit
  - I build my SQL procedures with this turned off
  - If I need to call my SQL procedure from another program or want to call it multiple times as a single transaction, use the IDMSIN01 utility to programmatically enable transaction sharing
- Transaction sharing is enabled
  - Only the top level or driver can issue a ROLLBACK
  - Be warned, if a lower level program does a ROLLBACK then the entire SQL command thread will fail

Future Direction

Future direction

- This has set the stage to convert the 3270 presentation layer to a modernized web presentation layer
- The 3270 and web versions, by design, coexist
- A completely modernized web front end, is possible with minimal effort and cost
- We already have a prototype using the same SQL procedures created for the current project
- We already have management buy in to move forward

Summary

- The CA ADS / SQL conversion is not trivial, it is not that difficult
- The process is repeatable
- This supports our current business requirement
- We are staged to support the future without the cost and time required to rewrite our system

Online Session Evaluation

Please provide your feedback about this session: A6

On the CA Communities web site: http://communities.ca.com

More details in your conference bag

Questions and Answers