May 6–10, 2007

San Jose Convention Center

San Jose, California, USA

Session: <F12>

Techniques for the Well-Dressed DB2 Stored Procedure

IDUG® 2007 North America

Dona Bell

large financial services company in the northern suburbs of Chicago

May 10, 2007 09:20 a.m. – 10:20 a.m.

Platform: <z/Os>

GoFurther

May 6–10, 2007

San Jose Convention Center

San Jose, California, USA

IDUG® 2007 North America

Techniques for the Well-Dressed DB2 Stored Procedure

Showcase your business stored procedures by dressing them for the runway with these techniques.

How do I do this with limited time and money?

GoFurther

Presenting today……

- Dona Bell, a seasoned veteran of multiple DBMS and industries, currently serving as a DB2 & SQL Server DBA at a large financial services company in the northern suburbs of Chicago for the last 12 years.
- Since she has a husband and three sons (blessedly, sons are on their own now), she likes to escape at times, and will relate how she connected the movie 'The Devil Wears Prada' to stored procedures.

Topics

- You can't design what you can't define!
- Accentuate the good qualities of your stored procedures
- Sideline bad designs; don't let them hit the runway
- Calling stored procedures by function
- DBAs & Applications designers have roles to play
- Techniques: good,bad,ugly: never accept bad or ugly!
- Coding Examples: the designer's racks
- Standards & References

What is a DB2 stored procedure (SP)?

- A specialized program executed by DB2 in response to an SQL CALL statement
- Both a database object, managed by DB2 & DBAs, like tables and indexes, and an application program that can access and/or modify data in any table in DB2 it is granted access to

What is a DB2 SP?

- Written in existing LE/370-supported programming languages, such as: Assembler, C, C++, COBOL, OO COBOL, PL/I, REXX and Java, as well as IBM's own extended SQL Procedure Language
- At my company, we invoke DB2 SP from DB2 Connect, CICS, Websphere, and batch mainframe jobs

What a SP isn't:

- An event-driven trigger (although a SP can be called by triggers, and call triggers itself)
- Required to be coded in a procedural SQL extension language – like other RDBMs (although DB2 does allow use of these languages from many and varied platforms)
- A simple I/O module

Accentuate the Positive

- Reusable code across multiple distributed platforms
- Consistent, easy to maintain, a high performer, safeguards data integrity

Accentuate the Positive

- Eliminates need for end user table authority and isolates processing
- 24 x 7 availability enhanced!
  - Stop stored procedures and queue requests while a new version is installed – then easily restart

Accentuate the Positive

- Removes client's dependence on database design at server
  - Less code changes at client locations
  - Dynamically change application programs

Accentuate the Positive

- In a client/server application:
  - Multiple SQL calls in a single unit of work reduces network traffic and improves performance
  - Better security; disallows client from manipulating dynamic SQL to read or update data that is out of scope

Accentuate the Positive

- In a non-client/server application:
  - Code reuse of a module executed from multiple clients, such as CICS, batch and/or Websphere without a static link of the stored procedure to each client's calling load module
  - Support of conditional logic in triggers & multiple triggers

Sideline bad SP designs

- Let's convert all these Oracle SP to DB2 as is!
- I have a series of subprograms nested six deep; I'll just convert these to SP!

Sideline bad SP designs

- Don’t assume your data security and good access paths will be enforced just by using a SP
- Avoid dynamic SQL in client server applications: you’re negating the benefit of keeping optimized static SQL at the server instead of the client

Sideline bad SP designs

- Match the needs of your business!
- At my company, we've sidelined SP calls from DB2 Connect for distributed applications where:
  - VSAM dataset access is required
  - You must call a CICS transaction from within
- If the above are true, CICS Transaction Gateway should be used instead of DB2 Connect/stored procedures

Calling SP by function

- Triggers that call DB2 stored procedures must use parameter style DB2SQL
  - IBM recommends parameter style DB2SQL for all SP because:
    - DB2SQL passes all parms from the SP to DB2
    - DB2SQL can return SQLSTATE to caller
    - DB2SQL modifies output parms and null indicators

Calling SP by function

- Avoid nesting SP calls (calling one SP from another SP):
  - Each SP holds a separate DB2 connection until commit, causing a shortage of connections
  - RESULTS SETS, if used, is negated for all but the first SP
  - A client calling SP1, which calls SP2 will also not get error diagnostic information from SP2

Calling SP by function

- When the calling program is CICS, or a Websphere client with two-phase commit, use COMMIT ON RETURN NO for SP definition.
- This is because the calling program must control the commits and not the SP.

Calling SP by function

- Calling a DB2 stored procedure (SP) from a CICS program:
- 1) DBA defines the SP using COMMIT ON RETURN NO
- 2) DBA must grant execute on the SP to the Bind Owner of the CICS calling program
- 3) Bind CICS calling program with PATH(schema), to reference the SCHEMA.SAMPLESP
- 4) Call SP SAMPLESP (must explicitly call SAMPLESP SP):
  - EXEC SQL
  - CALL SAMPLESP('ABC')
  - END-EXEC.

Application designer roles for SP

- Design points:
- Is my DBA engaged in the SP design?
  - Is remote access essential?
  - Who will call this SP? Will it call others?
  - What will Input and/or Output be?

Application designer roles for SP

- Further design points:
  - Are output result sets required? Specify max number of cursors that could be opened and returned to the client.
  - If there are no result sets (only output parms returned), DYNAMIC RESULT SETS should be set to 0 to reduce thread hold time.

Application designer roles for SP

- Further design points:
  - Frequency of execution? (STAYRESIDENT?)
  - Error processing in place?
  - Do NULL indicators need to be checked?
  - Is retry logic in place?
  - Triggers called? Or this SP called by one?
  - Is this a long-running SP? (ASUTIME?)

Application designer roles for SP

- Further design points:
  - If parameter style is GENERAL:
    - Make sure a trigger is not involved!
    - SP must set a return code
    - Was the SQLCODE of the CALL good?
    - Did SP complete its work?

DBA's design roles for SP

- Design points:
- Is the Application designer engaged in the SP design?

- WLM considerations: confirm placement & resources required (STAYRESIDENT,ASUTIME,MAIN,SUB)
- WLM considerations: STAYRESIDENT
  - NO for most SP
  - YES for high-volume executed SP if you have integrated the Refresh-WLM SP into your processing

DBA's design roles for SP

- Further design points:
- WLM considerations: ASUTIME
  - ASUTIME used to limit CPU consumption (see MVS Initialization & Tuning Guide (SC28-1751)):
    - Our standards:
    - 10000 for testing (slightly >1 CPU seconds)
    - 100000 for OLTP-type SP (<10 CPU seconds)
    - 2435000 for batch-type SP (We assign these to separate WLM environments as well.)

DBA's design Roles for SP

- Further design points:
- WLM considerations: PROGRAM TYPE MAIN,SUB
  - MAIN: higher CPU cost for call & cleanup, but guarantee resources are freed
  - SUB: lower CPU cost for call & cleanup (30K inst), however some languages restrict (PL/I),
    program MUST be dynamically loadable (fetch able), application must free resources

DBA's design roles for SP

- Further design points:
- Review SP against design standards, and then:
- Define the SP to DB2 (SP code provided by Application designer, or Procedural DBA group)

DBA's design roles for SP

- Further design points:
- Grant execute authority on that SP to the bind owner (DBPxxxx); used as owner in bind card of caller).
- At my company, programs calling DB2 SP must be bound with PATH(schema), where schema is the SP owner in DB2

DBA's design roles for SP

- Further design points:
- Have Application designer test run SP
- Reiterate design roles with other SP, looking at big picture with the Application designer

Coding Techniques: Good, Bad, Ugly - 1

- Good: SP achieves only one logical function or goal, uses CASE statements, avoids dynamic SQL, and calls only one nested SP
- Bad: tasks operate on the same input/output, does tasks in procedural order (batch jobstream-like), temporal tasks (startup/shutdown)
- Ugly: SP lists a choice of code, only one of which is selected in each pass, uses dynamic SQL, or does >1 nested call to other SP

Coding Techniques: Good, Bad, Ugly - 2

- Good: SP is defined with parameter style DB2SQL,input is validated to detect bad data,debug info from SQLCA is sent to output or a DB2 table to log errors to caller
- Bad: Abend information is missing SP name, timestamp, SQLCODE, SQLERRMC, program error table or processing message
- Ugly: SP calls nested SP, expecting information passed back from lowest level to initial caller, no debug info for DBA

Coding Techniques: Good, Bad, Ugly - 3

- Good: SP uses result sets (opens cursor), then returns control to client. DB2 will send data asynchronously in 32K blocks. When optimize for n rows (or fetch first n rows) coded in cursor, this blocking can be optimal
- Bad: SP uses result sets, but does not optimize in the cursor for n rows for the process it is performing.
- Ugly: SP attempts to control the cursor itself, and does not use result sets when called for

Caller invoking a SP w/result sets

```
EXEC SQL
01 LOCVAR SQL TYPE is
     RESULT-SET-LOCATOR VARYING.
 END-EXEC.
  .
 PROCEDURE DIVISION
  .
  MOVE value to PARM1.
  EXEC SQL
      CALL PROC(:PARM1)
  END-EXEC.
  EXEC SQL
       ASSOCIATE LOCATOR(:LOCVAR)
        WITH PROCEDURE PROC
  END-EXEC.
  EXEC SQL
     ALLOCATE C1 CURSOR
     FOR RESULT SET :LOCVAR
  END-EXEC.
  loop
   FETCH C1 into :DEPTNO;
```

Stored Procedure w/ result set

```
EXEC SQL
     DECLARE CSR1 CURSOR FOR
     SELECT DEPTNO
     FROM DEPT
     WHERE ADMRDEPT = :DEPTPARM
  END-EXEC.
    .
    .
    EXEC SQL
        OPEN CSR1
    END-EXEC.
    .
    .
     GOBACK.
```

Parameter Style General w/ Nulls

```
LINKAGE SECTION.
     ****************************************************************
     * INPUT & OUTPUT PARAMETERS
     ****************************************************************
      01 INVAR1                PIC S9(4) USAGE COMP.
      01 INVAR2                PIC X (15).
      01 OUTVAR3                 PIC X  (5).
     ****************************************************************
     * NULL INDICATORS
     ****************************************************************
     01 IND-INVAR1              PIC S9(4) USAGE COMP.
     01 IND-INVAR2              PIC S9(4) USAGE COMP.
     01 IND-OUTVAR3               PIC S9(4) USAGE COMP.
.
PROCEDURE DIVISION USING
      INVAR1,
      INVAR2,
      OUTVAR3,
      IND-INVAR1,
      IND-INVAR2,
      IND-OUTVAR3.
       .
```

Parameter Style General w/ Nulls

```
*************************************************************
* CHECK INPUT PARAMETERS FOR NULLS
*************************************************************

IF IND-INVAR1 < 0
   take action for null INVAR1
ELSE
   take action for non-null INVAR1.
IF IND-INVAR2 < 0
   take action for null INVAR2
ELSE
   take action for non-null INVAR2.
  .
*************************************************************
* SET NULL INDICATORS FOR OUTPUT PARAMETERS
*************************************************************
if OUTVAR3 has data
   MOVE 0 TO IND-OUTVAR3
else if OUTVAR3 is null
   MOVE -1 TO IND-OUTVAR3.
```

Parameter Style DB2SQL w/ Nulls

```
LINKAGE SECTION.
    ****************************************************************
    * INPUT & OUTPUT PARAMETERS
    ****************************************************************
     01 INVAR1                PIC S9(4) USAGE COMP.
     01 INVAR2                PIC X (15).
     01 OUTVAR3                 PIC X  (5).
    ****************************************************************
    * NULL INDICATORS
    ****************************************************************
    01 IND-INVAR1              PIC S9(4) USAGE COMP.
    01 IND-INVAR2              PIC S9(4) USAGE COMP.
    01 IND-OUTVAR3               PIC S9(4) USAGE COMP.
    ****************************************************************
    * DB2SQL PARAMETERS
    ****************************************************************
    01 LINK-SQLSTATE PIC X(5).
    01 LINK-PROC.
      49 LINK-PROC-LEN PIC 9(4) USAGE BINARY.
      49 LINK-PROC-TEXT PIC X(27).
    01 LINK-SPEC.
      49 LINK-SPEC-LEN PIC 9(4) USAGE BINARY.
      49 LINK-SPEC-TEXT PIC X(18).
    01 LINK-DIAG.
      49 LINK-DIAG-LEN PIC 9(4) USAGE BINARY.
      49 LINK-DIAG-TEXT PIC X(70).
```

Parameter Style DB2SQL w/ Nulls

```
PROCEDURE DIVISION USING
     INVAR1,
     INVAR2,
     OUTVAR3,
     IND-INVAR1,
     IND-INVAR2,
     IND-OUTVAR3
     LINK-SQLSTATE,
     LINK-PROC,
     LINK-SPEC,
     LINK-DIAG.
     .
     ************************************************************
     * CHECK INPUT PARAMETERS FOR NULLS
     ************************************************************
     IF IND-INVAR1 < 0
       take action for null INVAR1
     ELSE
       take action for non-null INVAR1.
     IF IND-INVAR2 < 0
       take action for null INVAR2
```

Parameter Style DB2SQL w/ Nulls

```
ELSE
  take action for non-null INVAR2.
    .
   ************************************************************
  * SET SQLSTATE WITH VALUE RETURNED AFTER SQL STATEMENT
  ************************************************************

  EXEC SQL
    ...
  END-EXEC.
  EVALUATE SQLCODE
    WHEN 0
      MOVE SQLSTATE TO LINK-SQLSTATE
    WHEN OTHER
      MOVE SQLSTATE TO LINK-SQLSTATE.
  ************************************************************
  * SET NULL INDICATORS FOR OUTPUT PARAMETERS
  ************************************************************
  if OUTVAR3 has data
    MOVE 0 TO IND-OUTVAR3
  else if OUTVAR3 is null
    MOVE -1 TO IND-OUTVAR3.
```

Common error codes

SQLCODE	Error Description	Error Resolution
+464	The number of result sets returned by the stored procedure exceeds the DYNAMIC RESULT SETS limit in its definition	Alter the stored procedure definition to specify the appropriate maximum number of result set cursors that can be returned to the client.
-430	Stored procedure has abended. When a stored procedure abends, DB2 puts that stored procedure in "Stop Abend" status and subsequent calls are rejected with SQLCODE –430.	Correct the error in the stored procedure, then START the procedure

Common error codes

SQLCODE	Error Description	Error Resolution
-440	Stored procedure could not be found	Possible causes: Stored procedure not defined CURRENT PATH value not set to stored procedure schema name IN/OUT parameters on CALL statement do not match parameters in stored procedure definition (SYSIBM.SYSPARMS)
-444	Stored procedure program could not be found	Possible causes: Load module not in stored procedures load library Load module copied from source library with different block size than stored procedures load library and not reblocked

Common error codes

SQLCODE	Error Description	Error Resolution
-450	Stored procedure overlaid storage beyond the declared length of an OUT parameter	Possible causes: VARCHAR OUT parameter defined, but maximum length differs between stored procedure code and stored procedure definition
-471	Stored procedure could not be invoked because of reason code specified in error message	Check reason code in DB2 Messages and Codes and correct.

Common error codes

SQLCODE	Error Description	Error Resolution
-551	Nested SP has called a SP without Execute granted to schema owner	Encountered after DB2 V 8 CM mode; fixed by: Grant schema name Execute on called SP Rebind calling SP
-751	A COMMIT or ROLLBACK was attempted, but not allowed for this stored procedure	COMMIT/ROLLBACK is not allowed when: The stored procedure is called by a client that uses 2-phase commit. This includes all CICS clients and can include Websphere clients when Websphere is configured to use 2-phase commit. The stored procedure is nested within a Trigger or User Defined Function

Common error codes

SQLCODE	Error Description	Error Resolution
-805	At execution time, DB2 could not find a Package in the Collection specified for the stored procedure with a precompile timestamp matching the load module's precompile timestamp	Possible causes: Package bound in wrong Collection Wrong Collection specified in stored procedure definition Package was recompiled but not rebound If stored procedure calls submodules, a submodule may have been recompiled but not rebound
-905	This stored procedure exceeded the CPU limit, as defined either by the stored procedure definition or by DB2's Resource Limit Facility (RLF)	If the "DERIVED FROM" clause of the error text specifies "SYSIBM.SYSROUTINES," the stored procedure exceeded the limit in the stored procedure definition; if it specifies "SYSIBM.DSNRLST01," the stored procedure exceeded the RLF limit.

Common error codes

SQLCODE	Error Description	Error Resolution
-905, cont.	This stored procedure exceeded the CPU limit, as defined either by the stored procedure definition or by DB2's Resource Limit Facility (RLF)	If the SYSROUTINES limit is being reached, reduce the CPU usage by tuning the SQL. If that doesn't work, alter the ASUTIME in the stored procedure definition to a higher value. If the RLF limit is being reached, reduce the CPU usage by tuning the SQL. The RLF limit applies system-wide and is a very high value, so a stored procedure that exceeds this limit would not be appropriate for an OLTP-type stored procedure and most likely would not be appropriate for a batch-type stored procedure either.
-925	A COMMIT was attempted because the stored procedure was defined as COMMIT ON RETURN, but is not allowed for this stored procedure	COMMIT ON RETURN is not valid for a stored procedure that is called by a client using 2-phase commit..

Common error codes

SQLCODE	Error Description	Error Resolution
-925, cont.	A COMMIT was attempted because the stored procedure was defined as COMMIT ON RETURN, but is not allowed for this stored procedure	This includes all CICS clients and can include Websphere clients when Websphere is configured to use 2-phase commit.
CICS AD2U	A COMMIT or ROLLBACK was issued by a CICS DAP	When using DAS, the CICS UOW starts in the DAS region, so the COMMIT/ROLLBACK should be issued from this region. The RQBLK-MAJ-RET-CODE field of the DAS Request Block is used to indicate whether a COMMIT or a ROLLBACK should be performed.

Standards and References

Standards: DB2 Engineering Website:

- DB2 Application Standards, along with lifecycle guidelines

References:

Joe Celko's 'SQL Programming Style'

Craig Mullins' 'DB2 Developer's Guide'

MVS Initialization & Tuning Guide (SC28-1751)

YL&A's 'DB2 High Performance Design & Tuning'

FINIS

- Any Questions?
- Email me at Belldonag@aol.com

Session: <F12>

Session Title: Well-Dressed DB2 Stored Procedure

Dona Bell

Belldonag@aol.com

GoFurther