

DB2 Stored Procedures – Trends and Technology

CATTERALL CONSULTING ☐☐☐

Robert Catterall, President

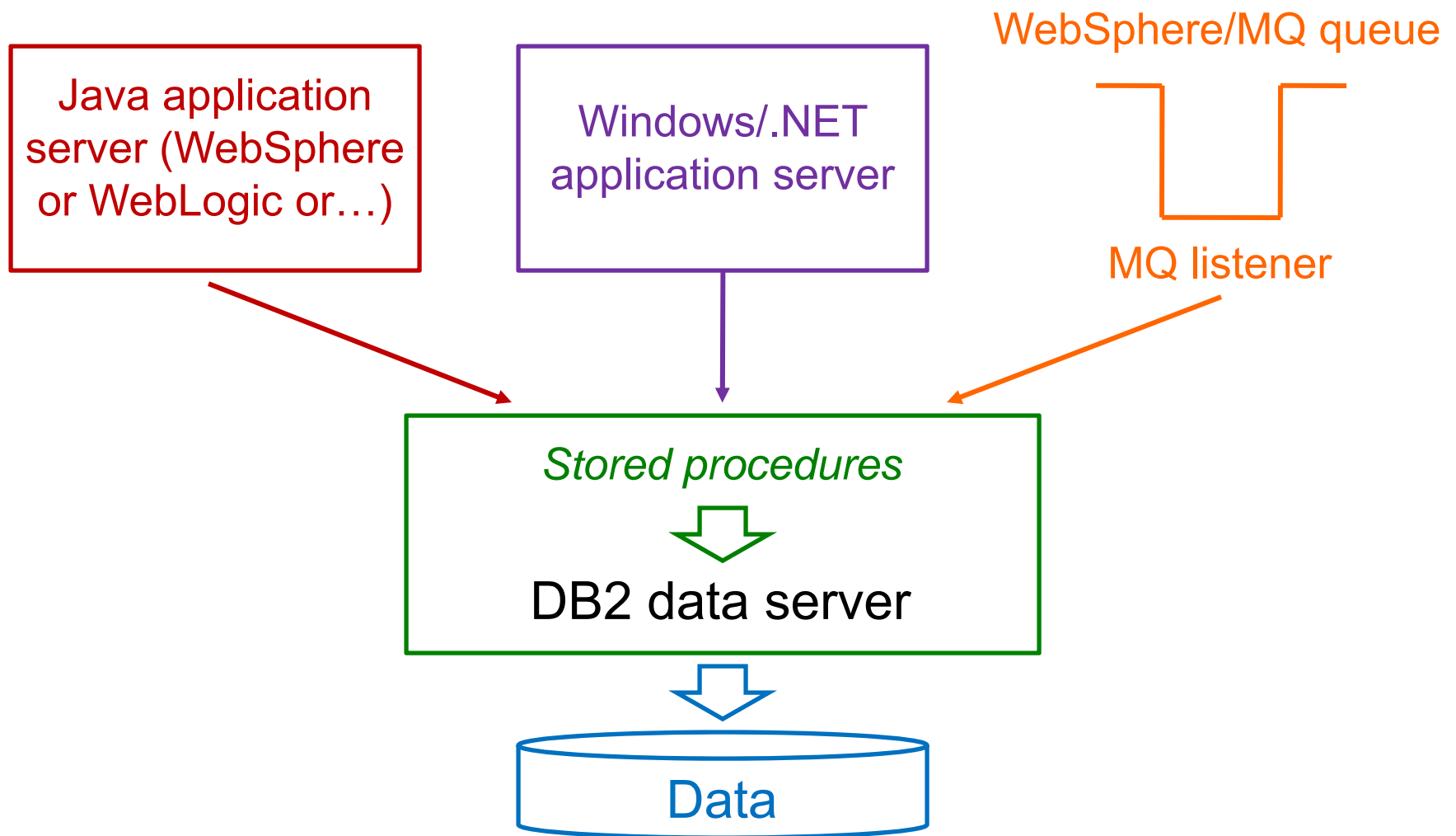
Michigan DB2 Users Group
March 12, 2009

Agenda

- “The vision thing”
- Advances in DB2 for z/OS stored procedure functionality since DB2 V4
- DB2 V9 native SQL procedures
- Moving forward

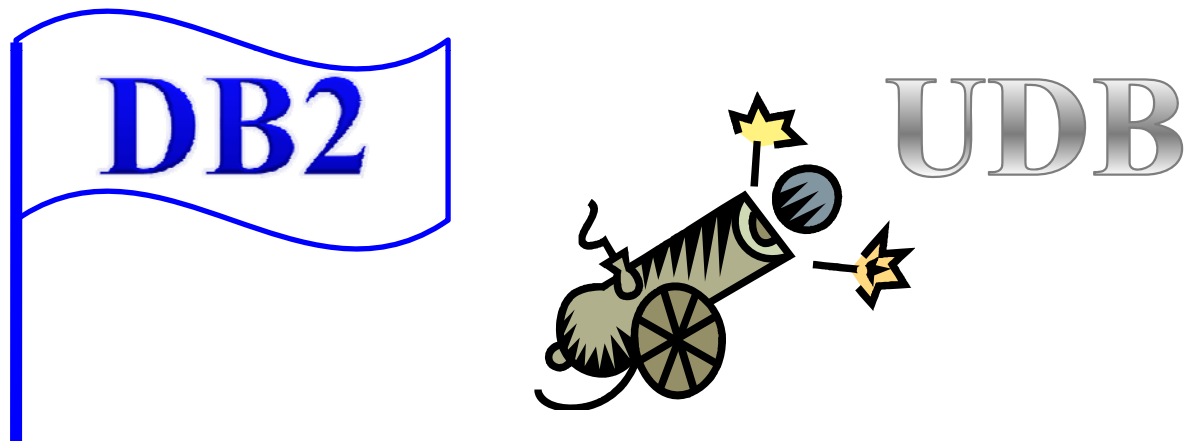
“THE VISION THING”

My vision of a DB2 data-serving system



The “vision” diagram: important points

- The DB2 server platform is not specifically identified – could be z/OS, or Linux, or UNIX, or Windows
 - Specific platform is not important: why treat them differently?



- It's not DB2 = mainframe and UDB = Linux/UNIX/Windows!
It's just DB2!!!

Vision diagram points (cont'd)

- The DB2 server is a pure database server – there is no transaction management subsystem on the server
 - This has been standard operating procedure for some time in the distributed systems world
 - Mainframes running DB2 often run CICS or IMS/TM, as well
 - Usually because the organization ran a DB2-accessing transactional workload before DB2 stored procedure functionality was available
 - Static, server-side SQL (important for scalability) can be packaged via CICS or IMS – or stored procedures*
 - Many mainframers believe that you have to have CICS or IMS/TM to support a high-volume DB2 for z/OS-based transactional workload
 - *I would contest that assumption, and I'm not the only one*

Vision diagram points (cont'd)

- MQ is a very important part of the picture
 - I believe that far too high a proportion of DB2-related application work is done in a synchronous manner
 - Introducing queues between layers of an application system (e.g., between application servers and data servers) can enhance system resiliency and improve end-user response time
 - The MQ listener, which can run on an application server or on the DB2 server, can automatically call stored procedures in response to certain messages being placed on an MQ queue
 - Could be a key component of workflow-orchestration functionality
 - Enables “transactionalization” of traditional batch work when it comes to handling input files (program places file records on a queue, and listener calls stored procedures to process records)

ADVANCES IN DB2 FOR z/OS STORED PROCEDURE FUNCTIONALITY SINCE DB2 V4

Ongoing functionality enhancements

- DB2 V4: stored procedure functionality introduced
 - *With rather limited functionality, to say the least* 😊
- DB2 V5: caller of a stored procedure can FETCH rows from a multi-row result set
 - Previously, output could only be returned via output parameters
 - Not useful if the number of result set rows could vary from one call of a stored proc to another
 - Even for a fixed-size result set, return of data via output parms unwieldy unless number of rows is very small

More enhancements...

- Also in DB2 V5: WLM-managed stored procedure address spaces (versus the old DB2-managed SPAS)
 - You could have several of these, vs. the one DB2-managed SPAS
 - WLM could automatically fire up additional instances of a stored procedure address space in response to workload demands
 - Stored procedures got two-phase commit capability via RRSAF (Recoverable Resource Services Attach Facility – required for stored procedures that execute in a WLM-managed address space)
 - Nice for things like updating DB2 and MQ via one stored procedure
 - You could run Java stored procedures in a WLM-managed address space
 - You could access a LOB (large object) using a stored procedure in a WLM-managed address space

And more...

- DB2 V6: CREATE/ALTER/DROP PROCEDURE statements added to DDL
 - Before that, DBA had to insert/update/delete rows in SYSPROCEDURES catalog table (tedious, error-prone)
- DB2 V7: SQL Procedure Language introduced
 - Stored procedures could be written entirely in SQL
 - SQL was extended to include logic flow-control statements such as GOTO, IF, ITERATE, LEAVE, LOOP, REPEAT, and WHILE
 - SQL procedure converted to C program under the covers, and executes like an external stored procedure
- Also in DB2 V7: COMMIT and ROLLBACK issued from stored procedure

And more...

- DB2 V8:abend limit can be set at individual stored proc level
 - Previously set for all stored procedures in a subsystem via a ZPARM parameter (STORMXAB)
 - If a stored procedure abends n times, it will be placed in stopped status
- Also in DB2 V8: better synergy with Workload Manager
 - WLM and System Resource Manager can recommend changes in the number of tasks that operate in a stored procedure address space
 - DB2 will then add or delete tasks based on WLM recommendations
 - NUMTCB for a WLM stored procedure application environment is upper limit on tasks in an address space instance

And...

DB2 V9: native SQL procedures!



(such a big deal, it gets its own section in this presentation)

What about DB2 for LUW?

- Haven't stored procedure capabilities been extended on this platform?
 - Of course they have – virtually everything one might do with a DB2 for z/OS stored procedure, one can do with a DB2 for LUW stored procedure
 - Exceptions to this rule would tend to involve mainframe-oriented things such as invoking a CICS transaction via stored proc
- Why the difference in emphasis?
 - Stored procedures have been a mainstay of DB2 for LUW enterprise-class application design for some time now
 - On the mainframe side, many DBAs and systems programmers (and some developers) are still wondering if they can build industrial-strength apps without a local transaction-manager component
 - They can (others have), and I aim to let them know that

DB2 V9 NATIVE SQL PROCEDURES

“This changes everything”

- **Before:** SQL stored procedure turned into a C language program under the covers
 - Runs like an external stored procedure (e.g., one that was initially written in C or COBOL versus SQLPL) in a WLM-managed stored procedure address space
 - Not-in-DB2 part of a C program consumes more CPU than does equivalent COBOL code (though less than Java)
- **Now:** native SQL procedure is just a package – a “runtime structure” based on the SQL statements to be executed
 - Native SQL procedure runs in the DB2 database services address space (aka DBM1)

Native versus external

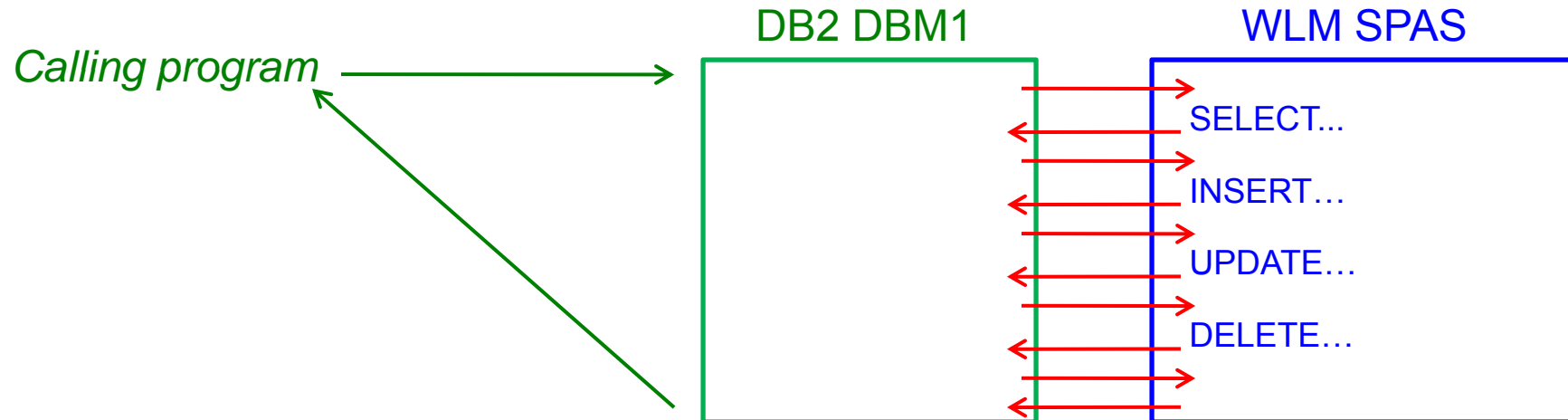
- External stored procedure: when called, runs under its own TCB
 - Caller's task (TCB or SRB) is suspended, and stored proc task uses caller's thread for communication with DB2
 - In some cases, there can be processing delays and a build-up of DBM1 virtual storage consumption associated with the switching of threads from calling tasks to stored proc tasks
- Native SQL procedure: everything runs under the caller's task – when the stored proc is called, the caller's DB2 thread just switches to the SP package
 - No queuing, no delays

A CPU efficiency benefit

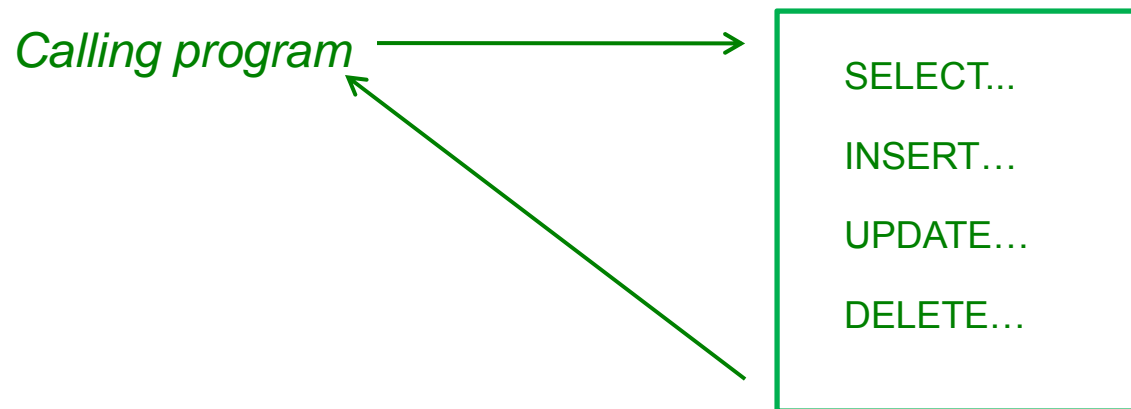
- For every SQL statement in an external stored procedure (or any other external-to-DB2, SQL-issuing program), an “addressability round trip” is required
 - Program’s task switches addressability from “home” address space (e.g., WLM-managed stored procedure address space) to DB2 DBM1 for SQL execution, then switches back
 - Utilizes z/OS “program call” (PC) functionality
 - Each round trip probably consumes a few thousand instructions, and that’s just the back-and-forth – not SQL execution in DBM1
- Native SQL procedures eliminate this extra path length
 - With the CALL to the native SQL procedure, you’re already in DBM1, and you stay there until the stored procedure completes

The picture of efficiency...

External SQL procedure



Native SQL procedure



The zIIP factor

- Refers to the z9 Integrated Information Processor, a specialty mainframe engine that can run eligible workloads and which does not factor into mainframe software pricing
- A native SQL procedure is zIIP-eligible if it is invoked via a remote call through the DB2 Distributed Data Facility (DDF)
 - Why restricted to remote vs. local CALLs? Because IBM is willing to leave mainframe software money on the table if you will move your mainframe DB2 system closer to the Vision depicted on slide 4
(see, it's not just my vision)
- In benchmark tests, the amount of CPU processing directed to a zIIP engine has exceeded 50% for some native SQL procedures

More good stuff: functionality

- **Nested compound statements**
 - A compound statement is a grouping of SQL statements, bounded by BEGIN and END
 - Within a compound statement, variables, cursors, and condition handlers can be declared
 - Now that SQL stored procedures can contain nested compound statements, condition handlers can have their own compound statements
 - Benefit: enables significantly more sophisticated error handling within SQL stored procedures
- **Better compatibility across the DB2 Family**
 - Much easier to port native DB2 SQL procedure to DB2 systems running on Linux, UNIX, or Windows servers

And more: lifecycle simplification

- Re: stored procedure lifecycle: create, manage, maintain
- No DB2-external resources (e.g., source, object, and load libraries) and processes (e.g., compile and link) to deal with
 - Among other things, that means a less-involved security and authorization set-up for stored procedure creation (vs. external SP)
 - Consider the SECURITY specification (USER vs. DB2) for the IBM-supplied external SQL procedure-building routine DSNTPSMP, and the authorizations needed to run that routine (including catalog table access)
 - Fewer worries about program/package coordination (-805 SQL code)
- Note: “simpler” does not always mean “easier”
 - Which is “easier”: Spanish or English? [What do you speak?]



More on this to come...

Native SQL or COBOL?

Native SQL stored procs	COBOL stored procs
<ul style="list-style-type: none">• zIIP-eligible when called via DRDA• Simplified versioning• Easier to build and deploy• Improved cross-platform portability (not just different DB2 platforms – Oracle, SQL Server, etc.)• Doesn't require COBOL skills (not a problem if you've got 'em)	<ul style="list-style-type: none">• Not zIIP-eligible (unless it issues a parallelized query)• Program version has to be coordinated with package version• Good way to utilize in-house COBOL skills (if you've got 'em)• Allows for greater sophistication re: logic, functionality<ul style="list-style-type: none">– How much sophistication do you need in your data layer?


MOVING FORWARD

Getting from here to there...

- “There” being a situation in which you’re getting maximum benefit from the use of DB2 stored procs
- While still on DB2 for z/OS V8:
 - USE STORED PROCEDURES (if not already doing so)!
 - Even if COBOL is your preferred stored procedure programming language in the DB2 V8 environment, code and deploy some SQL stored procedures to gain familiarity (again, if you’ve not already done so)
 - I believe that native SQL procedures are the way of the future, and it would be a good idea to get ready for that future



While still on V8 (if you use CICS)

- At least in development/test (ideally in production), make the functionality of one or more CICS-DB2 transactions available to DRDA requesters via stored procedure calls
 - A more “open” way to expose the transaction functionality
- Could be done by converting COBOL CICS program to COBOL stored procedure (usually involves very little change), or replicating functionality in SQL procedure, or **using stored procedure to invoke CICS transaction** 
 - One option: DSNACICS stored procedure that comes with DB2
 - Alternative: code to the CICS EXCI interface yourself, using either:
 - EXCI CALL, or
 - EXEC CICS (easier to code, more frequently used)
(analogous to DB2 TSO attach versus CALL attach)

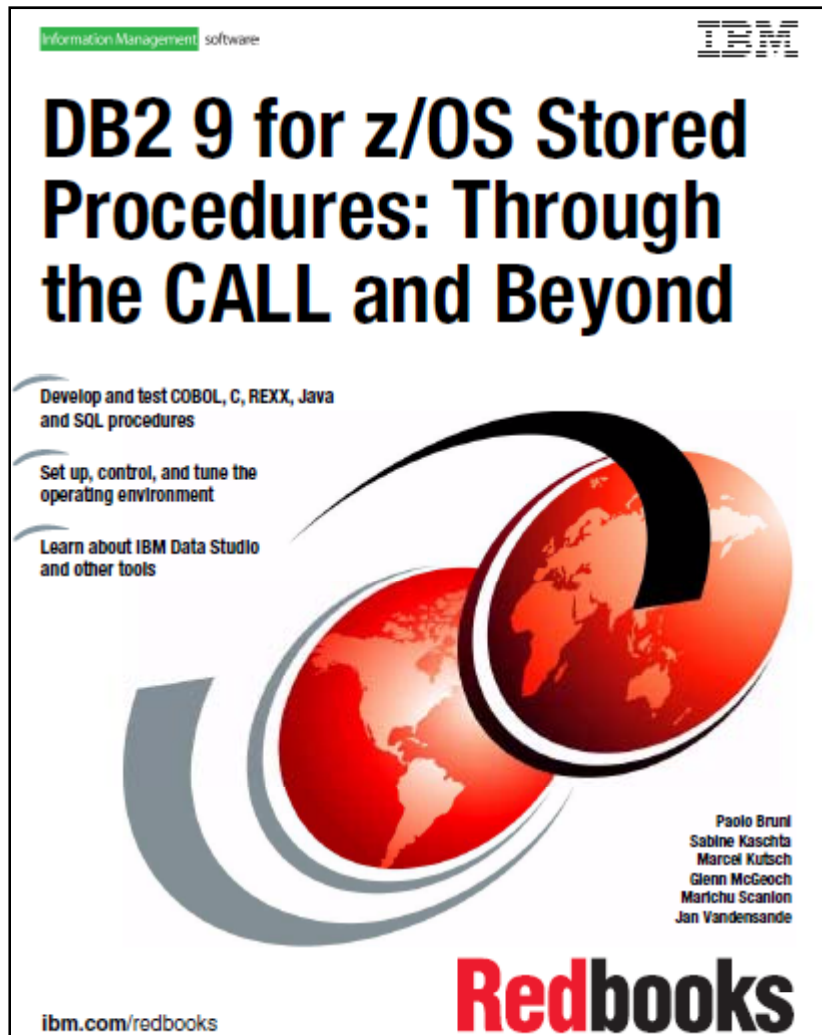
When you go to DB2 V9...

- If you don't already have zIIP engines, think seriously about getting some (or maybe adding to what you have)
 - I strongly suspect that we'll continue to see IBM make more work zIIP-eligible with future releases of DB2
- Use native SQL stored procedures (once you're in DB2 V9 New Function Mode), *but be deliberate about this if you're already using external SQL procedures*
 - The simpler lifecycle processes of native SQL procedures are less compelling if your external SQL procedure infrastructure is well established and mature
 - Bottom line: the advantages of native SQL procedures versus external SQL procedures make conversion worthwhile, *but you'll want to do that in a non-disruptive fashion*

If already using external SQL procs

1. Get familiar and comfortable with the different lifecycle processes associated with native SQL procedures
2. Consider converting external SQL procedures to native SQL procedures when upgrading an existing application
 - Not difficult – basically a matter of dropping the existing procedure and re-executing CREATE PROCEDURE without the FENCED and EXTERNAL options
 - Also, remove WLM ENVIRONMENT specification, if present (you can have WLM ENVIRONMENT FOR DEBUG MODE, and you need this if you want to run the proc in debug mode and you don't have a default WLM environment specified via ZPARM WLMENV)
3. As appropriate, choose native SQL procedures for new DB2 application development

A great documentation resource



- A complete update of a classic IBM “red book” first published in 2004 for DB2 for z/OS V7
- To download PDF (for free):
 - Go to www.ibm.com/redbooks
 - Search on SG24-7604

Thank You

Robert Catterall

Catterall Consulting

rcatterall@catterallconsulting.com