DB2 for z/OS
Native SQL Procedures: User Experiences in Implementation

Linda F. Claussen
Themis, Inc.
lclaussen@themisinc.com
Topics Covered

• Native SQL procedure development Lifecycle
• User Implementation Experiences
• Interfacing with change management tools
• Critical performance issues
• What’s new in DB2 10
• Recap
What are Stored Procedures

- User-written SQL modules stored at a DB2 Server
- Can be invoked by Client Applications
  - Local client applications
  - Remote DRDA client applications
  - Remote private protocol client applications
- Client can pass parameters to the stored procedure and receive parameters and/or result sets from the stored procedure
Why use Stored Procedures?

Client

EXEC SQL ...

EXEC SQL ....

EXEC SQL FETCH 100 times :

EXEC SQL ....

Presentation Logic + Business Logic

Server

Perform SQL ...

Perform SQL ....

Perform SQL FETCH 100 times :

Perform SQL ....

Data Management
Why use Stored Procedures?

Client

EXEC SQL CALL...

....

:

Process ....

Presentation Logic

Server

EXEC SQL ...

EXEC SQL ....

EXEC SQL FETCH 100 times :

EXEC SQL ....

GOBACK

Business Logic

+ Data Management
When?

- Reduce Network Traffic
- Maintain a single copy of Business Logic
- Ease of Maintenance
- Prevent Client from manipulating contents of sensitive server data
- Access features only available at the server
- Exploit computing power of the server
- Access server resources
Types of Stored Procedures

- **External**
  - COBOL
  - C, C++
  - PL/I
  - Assembler
  - REXX
  - Java
  - SQL PL

- **Native**
  - SQL PL

New in DB2 9 for z/OS
CONNECT TO ssid
CALL SP1 (val1,val2, …)
Native Stored Procedures

CONNECT TO ssid
CALL SP1 (val1,val2, ...)

z/OS

ssidDIST

DB2TCP/IPInterface

Data

ssidDBM1

SP1Code

CONNECT TO ssid
CALL SP1 (val1,val2, ...)

z/OS

ssidDIST

DB2TCP/IPInterface

Data

ssidDBM1

SP1Code
Coding a SQL PL Procedure

• An SQL procedure consists of:
  - CREATE PROCEDURE header
  - BEGIN statement
  - Body (SQL procedural statements and / or DB2 SQL statements)
  - END statement

• Comments within an SQL procedure:
  - -- for a single line comment
  - /* to start */ end multiple-lines comments

• Statements end with semicolon
A SQL PL Header

CREATE PROCEDURE SPA80 (IN p_DNO CHAR(3), OUT p_CNT SMALLINT, OUT p_SUMSAL DECIMAL(11,2), OUT p_SQLCODE INTEGER)

Parameters

Version V1

- Procedure Name
  - 128 byte max length
  - Unique within Schema / Collection
  - Schema / Collection ID will be supplied when create is deployed
- Parameters
  - 128 byte max length
  - Can be IN bound, OUT bound or INOUT (both directions)
  - Used to pass data between procedure and caller
  - Cannot specify a default value
- Versioning
  - 64 EBCDIC bytes max length
  - If using versioning do not use the default V1 naming convention
A SQL PL Header

LANGUAGE SQL
-- COMMON BIND OPTIONS
CALLED ON NULL INPUT
RESULT SETS 0
QUALIFIER THEMIS1
PACKAGE OWNER DBTHM40
ASUTIME LIMIT 500000
COMMIT ON RETURN NO
CURRENT DATA NO
DEGREE ANY
WITHOUT EXPLAIN
ISOLATION LEVEL CS
VALIDATE BIND

...
P1: BEGIN
SET p_CNT = 0;
SET p_SUMSAL = 0;
SET p_SQLCODE = 0;
SELECT COUNT(*), SUM(SALARY)
  INTO p_CNT, p_SUMSAL
  FROM EMP;
END P1
SQL Procedure Statements

- DECLARE Statement
- Assignment Statement
- CALL, GOTO, LEAVE, RETURN
- IF, CASE, WHILE, LOOP, REPEAT, ITERATE, FOR
- Compound statement
- GET DIAGNOSTICS statement
- SIGNAL, RESIGNAL statements
- SQL Statements

- Note: Successful Execution of any SQL statement will set SQLCODE variable value to 0 and SQLSTATE variable value to ‘00000’.
CREATE PROCEDURE SPA80 (OUT p_CNT1 SMALLINT
,OUT p_SUMSAL DECIMAL(11,2)
,OUT p_SQLCODE INTEGER
)

VERSION V1
ISOLATION LEVEL CS       VALIDATE BIND
PACKAGE OWNER DBTHM80    QUALIFIER THEMIS1
RESULT SETS 0             LANGUAGE SQL

P1: BEGIN
    DECLARE SQLCODE INTEGER DEFAULT 0;

    SELECT COUNT(*), SUM(SALARY)
    INTO p_CNT1, p_SUMSAL
    FROM EMP;

    SET p_SQLCODE = SQLCODE;
END P1
CREATE PROCEDURE SPB80  (OUT p_SQLCODE INTEGER)
  VERSION V1
  ISOLATION LEVEL CS
  VALIDATE BIND
  PACKAGE OWNER DBTHM80
  QUALIFIER THEMIS1
  RESULT SETS 1
  LANGUAGE SQL

P1: BEGIN
  DECLARE SQLCODE INTEGER DEFAULT 0;

  DECLARE CURSOR1 CURSOR WITH RETURN FOR
  SELECT EMPNO, LASTNAME, MIDINIT, FIRSTNME,
       SALARY, DEPTNO
  FROM EMP
  ORDER BY DEPTNO, EMPNO;

  OPEN CURSOR1;
  SET p_SQLCODE = SQLCODE;
END P1
CREATE PROCEDURE SPXNER ( IN DEPTIN CHAR(3)
    ,OUT LINE_NUM   INTEGER
    ,OUT REASON_CODE INTEGER
    ,OUT SQLCODE_OUT INTEGER
    ,OUT ROW_NUM     DECIMAL(31,0)
    ,OUT MSG_OUT  VARCHAR(132) )
~
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        LINE_NUM   = DB2_LINE_NUMBER,
        REASON_CODE = DB2_REASON_CODE,
        SQLCODE_OUT = DB2_RETURNED_SQLCODE,
        ROW_NUM     = DB2_ROW_NUMBER,
        MSG_OUT     = MESSAGE_TEXT;
    END;
Example with Logic

CREATE PROCEDURE SPXNLC1 ( IN QID CHAR(1), OUT RETCODE INTEGER)

VERSION V1
ISOLATION LEVEL CS
QUALIFIER DB1029TB
RESULT SETS 1
LANGUAGE SQL

PM: BEGIN
DECLARE SQLCODE INTEGER DEFAULT 0;
SET RETCODE = 0;
IF QID = 'R' THEN GOTO P2;
ELSEIF QID = 'E' THEN GOTO P3;
ELSEIF QID = 'D' THEN GOTO P4;
ELSE GOTO PEXIT;
END IF;
P2: BEGIN
DECLARE cursor1 CURSOR WITH RETURN FOR
SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES;
OPEN cursor1;
GOTO PEXIT;
END P2;

P3: BEGIN
DECLARE cursor2 CURSOR WITH RETURN FOR
SELECT LASTNAME, SALARY FROM EMP;
OPEN cursor2;
GOTO PEXIT;
END P3;

P4: BEGIN
DECLARE cursor3 CURSOR WITH RETURN FOR
SELECT ADMRDEPT, DEPTNAME, LOCATION, EMPNO, DEPTNO FROM DEPT;
OPEN cursor3;
GOTO PEXIT;
END P4;

PEXIT: BEGIN
SET RETCODE = SQLCODE;
END PEXIT;
END PM
Example #2

IF QID = 'R' THEN
    BEGIN
        DECLARE cursor1 CURSOR WITH RETURN FOR
            SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES;
        OPEN cursor1;
    END;
ELSEIF QID = 'E' THEN
    BEGIN
        DECLARE cursor2 CURSOR WITH RETURN FOR
            SELECT LASTNAME, SALARY
            FROM EMP;
        OPEN cursor2;
    END;
ELSEIF QID = 'D' THEN
    ~
ELSE GOTO PEXIT;
Example #3

P3: BEGIN

DECLARE cursor2 CURSOR WITH RETURN FOR
SELECT LASTNAME, SALARY
FROM EMP;

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
    GET DIAGNOSTICS CONDITION 1 MSG_OUT = MESSAGE_TEXT;
    SET RETCODE = SQLCODE;
END;

~

OPEN cursor2;
GOTO PEXIT;
END P3;
Development Life Cycle

• CREATE PROCEDURE
  – Data Studio
  – SPUFI, DSNTEP2, etc.

• ALTER ADD/REPLACE/ACTIVATE VERSION

• BIND PACKAGE ~ DEPLOY

• REBIND package vs. REGENERATE procedure
Using SPUFI

• Set SPUFI preferences:
  --#SET SQLFORMAT SQLPL
  --#SET TERMINATOR @

• Set the Schema/Collection ID
  SET CURRENT SCHEMA = ‘DB1029CL’@

• Code your Create statement
  CREATE PROCEDURE
  ~
  P1: BEGIN
  ~
  END P1@
Using DSNTEP2

//SYSTSIN DD *
DSN SYSTEM(D91A)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP91) +
LIB('DSN910.RUNLIB.LOAD') +
PARMS('/SQLFORMAT(SQLPL),SQLTERM(@)')
END
//*
//SYSSIN DD *
CREATE PROCEDURE
IBM Data Studio®

• Create Procedure
  – Wizard with user defined fragments
  – Templates
  – Drag & Drop
  – Open data source with Routine Editor
  – Copy / Paste
  – Import / Export

• Deploy

• Run / Test (project view or source)

• Debug
Define a Native SQL Procedure

CREATE PROCEDURE DBTHM.SP3N
PACKAGE OWNER ownerid
VERSION SP3N_VR1
QUALIFIER THEMIS90

DB2 CATALOG
SYSROUTINES
SYSROUTINESTEXT
SYSPARMS
SYSPACKAGE
SYSPACKSTMT
SYSPACKDEP
SYSENVIRONMENT

DB2 DIRECTORY
SPT01
SPT
SP1N
Considerations for Native SPs

- Procedural code and data access SQL are bound into the same package
- Larger packages mean:
  - More EDM Pool (memory) consumed
  - Larger SPT01 (disk) in the directory
  - Text of Create stored in SYSROUTINETEXT
**DB2 9 Directory**

```
DSNT360I -9A ***********************************************
DSNT361I -9A * DISPLAY DATABASE SUMMARY
       *   GLOBAL
DSNT360I -9A ***********************************************
DSNT362I -9A DATABASE = DSNDB01  STATUS = RW
       DBD LENGTH = 14200

DSNT397I -9A
NAME   TYPE PART  STATUS            PHYERRLO PHYERRHI CATA
-------- ---- ----- ------------------              --------
DBD01   TS      RW
SPT01   TS      RW
SCT02   TS      RW
SYSUTILX TS      RW
SYSLGRNX TS      RW
DSNSCT02 IX     RW
DSNSPT01 IX     RW
DSNSPT02 IX     RW
DSNLUX01 IX     RW
DSNLUX02 IX     RW
DSNLLX01 IX     RW
DSNLLX02 IX     RW

******* DISPLAY OF DATABASE DSNDB01  ENDED  ************

DSN9022I -9A DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```
### Display of Database Summary

The database `DSNDB01` is in read/write (RW) status with a DBD length of 108200.

#### Table of Database Names and Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>STATUS</th>
<th>PHYERRLO</th>
<th>PHYERRHI</th>
<th>CATALOG</th>
<th>PIECE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD01</td>
<td>TS</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBD01</td>
<td>TS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPT01</td>
<td>TS</td>
<td>0001</td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPT01</td>
<td>TS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCT02</td>
<td>TS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSUTILX</td>
<td>TS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSLGRNX</td>
<td>TS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSDBDXA</td>
<td>LS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSSPUXA</td>
<td>LS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSSPUXB</td>
<td>LS</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNSCT02</td>
<td>IX</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNSPT01</td>
<td>IX</td>
<td>L0001</td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNSPT01</td>
<td>IX</td>
<td></td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNSPT02</td>
<td>IX</td>
<td>L0001</td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Display of Database DSNDB01 Ended**
## DBM1 Storage Relief

<table>
<thead>
<tr>
<th></th>
<th>DB2 9</th>
<th>DB2 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skeleton Pool</td>
<td></td>
<td>Skeleton Pool</td>
</tr>
<tr>
<td>Global Stmt Pool</td>
<td></td>
<td>Global Stmt Pool</td>
</tr>
<tr>
<td>DBD Pool</td>
<td></td>
<td>DBD Pool</td>
</tr>
<tr>
<td>EDM Pool</td>
<td></td>
<td>EDM Pool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>2g</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>EDMPOOL Working memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Working memory**

---

**Themis Inc.**
DB2 10 Performance Enhancements

- **Execution**
  - D91A:
    DB1029CL.SP4E00 - Execution Time => 4 s: 41 ms
  - DA1A:
    DB1029CL.SP4E00 - Execution Time => 1 s: 279 ms

- **Package Size**
  - D91A:
    DB1029CL SP5N00 PKSIZE=5208 AVGSIZE=16220
  - DA1A:
    DB1029CL SP5N00 PKSIZE=5192 AVGSIZE=17748
Naming Conventions

CREATE PROCEDURE
DBTHM.SP3N
PACKAGE OWNER
ownerid
VERSION VR1
QUALIFIER THEMIS90

SYSROUTINES
- NAME: SP3N
- SCHEMA: DBTHM
- OWNER: ownerid
- COLLID: DBTHM
- VERSION: VR1
- TYPE: N
- CONTOKEN: X’20’

SYSPACKAGE
- NAME: SP3N
- COLLID: DBTHM
- VERSION: VR1
- TYPE: N
- QUALIFIER: THEMIS90
Using Explain

LANGUAGE SQL
-- COMMON BIND OPTIONS
CALLED ON NULL INPUT
RESULT SETS 0
QUALIFIER THEMIS1
PACKAGE OWNER DB1029
ASUTIME LIMIT 500000
COMMIT ON RETURN NO
CURRENT DATA NO
DEGREE ANY
WITH EXPLAIN
ISOLATION LEVEL CS
VALIDATE BIND
...

Plan Table Qualifier
EXPLAIN YES
### Versioning

**SYSROUTINES:**

<table>
<thead>
<tr>
<th>SCHEMA</th>
<th>NAME</th>
<th>COLLID</th>
<th>VERSION</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBTHM</td>
<td>SP3N</td>
<td>DBTHM</td>
<td>VR1</td>
<td>Y</td>
</tr>
<tr>
<td>DBTHM</td>
<td>SP3N</td>
<td>DBTHM</td>
<td>VR2</td>
<td>N</td>
</tr>
</tbody>
</table>

**ALTER PROCEDURE DBTHM.SP3N**

ADD VERSION VR2

~

SELECT COUNT(*)

INTO DCOUNT

FROM EMP

WHERE DEPTNO >= DNUM;

END P1 #

**ALTER PROCEDURE DBTHM.SP3N**

ACTIVATE VERSION VR2

**ALTER PROCEDURE DBTHM.SP3N**

DROP VERSION VR1
Versioning

ALTER PROCEDURE DBTHM.SP3N
REPLACE VERSION VR2
    ( IN DNUM CHAR(3)
    , OUT DCOUNT SMALLINT
    , OUT SUMSAL DEC(11,2)  )
~
SELECT COUNT(*), SUM(SALARY)
    INTO DCOUNT, SUMSAL
FROM EMP
WHERE DEPTNO = DNUM;

* DSNT408I SQLCODE = -20314, ERROR:
Versioning

ALTER PROCEDURE DBTHM.SP3N
    REPLACE VERSION VR2
    ( IN  DNUM  CHAR(3)
      , OUT DCOUNT SMALLINT )
    ISOLATION LEVEL UR
SELECT COUNT(*)
    INTO DCOUNT
    FROM EMP
    WHERE DEPTNO = DNUM;
END P1 #

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
Regenerate

- ALTER PROCEDURE DBTHM.SP3N REGENERATE ACTIVE VERSION
  - Rebinds Control Statements and SQL at local server
  - You must explicitly use the BIND PACKAGE COPY for all remote servers

- REBIND PACKAGE
  - Rebinds SQL only
DEPLOY

- **BIND PACKAGE(CHICAGO.DBPROD)
  DEPLOY(DBTHM.SP3N) COPYVER(VR2)
  ACTION(ADD) QUALIFIER(THEMISPD)**
DB2 9 New Special Registers

- SET CURRENT ROUTINE VERSION =
  - routine-version-id
  - host-variable
  - string-constant

- SET CURRENT DEBUG MODE =
  - host-variable
  - DISALLOW
  - ALLOW
  - DISABLE
DB2 10 Enhancements

- Define a parameter as a built-in or distinct type
- Define SQL param or variable as XML data type
- Limited use of scrollable cursors
- PLANMGMGT support added
- Support for multiple assignments in one SET statement:
  
  SET SQL_variable = expression,
  SQL_parm = expression ;
User Experiences

- Identify Limitations & Strengths
- Develop Best Practices guidelines
- Change Management Tools?
Native SQL Procedures

Strengths
- Easy & fast
- Tools
- Little Server knowledge Required
- One step Creation
- Portable
- Do not run in WLM
- zIIP eligible DDF $$$$
Best Practices

- Field of Use
- Development Tools
- Naming Conventions
- Common Error Handling Guidelines
- Common SP Shells/Templates
- Change Management Procedures
Development Tools

• Data Studio used for Unit Level Development, Debugging, Tuning & Testing

• DBA or Change Management Tool interface promotes to next level (QA, Production)

• Procedure you choose should be well documented
Field of Use

1. Data Studio for Development in Unit testing environment only

2. Query processing only – No updates
   Reasons: Error logging, lack of procedural logic, lack of experience, unit of recovery concerns

3. Minimal to moderate procedural logic (DB2 9)
   Use Cobol SPs when complex procedural logic req.
   Reasons: lack of procedural logic, performance of obj. code, size of package

4. SQL can be as complex as you wish
Common Naming Conventions

• Stored Procedure Name
  – @@@N#### (1st 8 positions unique)
  – @@@ application prefix
  – N for Native SQL Procedure
  – #### 4 position unique number

• Version ID
  – SP Unique Name with _V## suffix (@@@N####_V##)
  – Appl. Prefix with _V## suffix (@@@N_V##)

• All Parameters prefixed with p_
• All Declared Variables prefixed with v_
Common Error Handlers

• Return detail to Caller
• Insert into an Error Table
• Call a common Cobol Stored Procedure Error Handler
• Any combination of the above
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
GET DIAGNOSTICS CONDITION 1
LINE_NUM = DB2_LINE_NUMBER,
REASON_CODE = DB2_REASON_CODE,
SQLCODE_OUT = DB2_RETURNED_SQLCODE,
ROW_NUM = DB2_ROW_NUMBER,
MSG_OUT = MESSAGE_TEXT;
END;
DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1

ROLLBACK;

INSERT INTO ERRRTBL
(MODNAME, E_LINE_NUM, E_REASON_CODE,
 E_SQLCODE_OUT, E_ROW_NUM, E_MSG_OUT)
VALUES('SPXNER2', LINE_NUM, REASON_CODE,
       SQLCODE_OUT, ROW_NUM, MSG_OUT);

COMMIT;

SET LOG_SUCCESS = SQLCODE;

END;
Call a Common Error Handler

DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
GET DIAGNOSTICS CONDITION 1
V_LINE_NUMBER   = DB2_LINE_NUMBER,
V_REASON_CODE = DB2_REASON_CODE,
P_SQLCODE_OUT = DB2_RETURNED_SQLCODE,
V_ROW_NUMBER     = DB2_ROW_NUMBER,
P_MESSAGE_TEXT = MESSAGE_TEXT,
V_LINE_NUMBER   = DB2_LINE_NUMBER,
V_REASON_CODE = DB2_REASON_CODE,
CALL DB1029CL.SPXNER5   (V_NAME, V_LINE_NUMBER,
P_SQLCODE_OUT, V_REASON_CODE, V_ROW_NUMBER,
P_MESSAGE_TEXT, V_LOG_SUCCESS);
SET V_LOG_SUCCESS = SQLCODE;
END;
Templates & Shells

• Data Studio
  – 2.2.0 you can setup fragments for the SP Wizard
  – 2.2.1 you can setup templates for the SP Wizard

• Code a common Native SQL Procedure Shell
  – Usable in all environments
  – Should include comments to direct content
Change Management

- Develop Exit that invokes DSNTEP2
- Develop procedures to port Native SQL procedure DDL into a PDS
  - Change request to DBA, DBA Deploys to next level using Data Studio
  - Export to a text file & FTP to host PDS
  - Export to a text file & email as attachment to your DBA
  - Vendor tool SHOW DDL & save to PDS
CALL SP3N

Procedure Execution

CALL SP3N

DB2 DBM1

EDM

SP3N: SQL PL native logic

SQL

SQL

DB2 DIRECTORY

SPT01

DB2 CAT

SYSROUTINES
SYSPARMS
SYSPACKAGE
SYSENVIRONMENT

CALL SP3N

DDF

Themis inc.
Performance

• Your Code!!
• Network Issues
• DB2 Issues
  ➢ Internals
  ➢ DDF

Network & DB2 Connect

DDF

DB2 DBM1

SHR zIIP?

WLM

Themis inc.
Review Your Policies & Procedures

- Naming Conventions
  - Schema
  - Collection ID
- Create Procedure Authorization
- Common Error Handlers
- Use of Temp Tables
- Deployment / Propagation Procedures
- REBIND/REGENERATE Procedures
- EXPLAIN
User Reports on DB2 9

- zIIP usage was as advertised when calling via DDF, average 40% redirect
- Short running SQL Procedures 40 - 60% ITR (Internal Throughput Ratio), long running little or no improvement.
- Significant Development Time Savings
- Offload development to Front End or Mid-tier developers
Recommended Reading

• Redbook – DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond
• Best practices when using Data Studio and Optim Development Studio with DB2 for z/OS
• Doc: TD104524
  http://www.ibm.com/support/techdocs/