

What's so exciting about DB2 Native SQL Procedures?

This is a question I've been asked countless times. I can't help it, they excite me. To me they truly represent the future of computing. To put it in perspective think about your current development environment. How many lines of application code are you attempting to support with fewer and fewer experienced developers to maintain that code? Have you tried lately to hire a developer that knows COBOL, JCL, TSO, VSAM, or any of the skills it takes to develop traditional mainframe applications? Try doing a search on application development; know what comes up? " Experts in Web, Mobile and Desktop development" or "certified PHP, java, .Net web developers" and don't forget " RAD, Rapid application development that uses minimal planning in favor of rapid prototyping". Well, the problem is my data lives, quite happily I might add, on an IBM System Z in DB2 databases. Why you might ask? It handles the terabytes of data required efficiently, securely, 24/7 and with full recoverability. Now, how to continue providing access to that information anywhere, anytime with minimal overhead and cheaper than ever before? Thus, Native SQL Procedures EXCITE ME!

Now, let's take a look at DB2 SQLPL. DB2 SQL Procedural Language (SQL PL) is a subset of the SQL Persistent Stored Modules (SQL/PSM) language standard. This standard is the basis for the structured programming languages used with SQL to write stored procedures, functions, triggers, and standalone code. Encapsulating complex business logic in database stored procedures can yield the following benefits:

- Significantly improved application performance
- Increased application scalability
- Simplified application development
- Reduced network traffic

The SQL/PSM standard combines the ease of SQL data access with the flow control structures of a simple programming language. It gives developers the ability to create compound SQL statements and procedures that only need to be coded once to run on multiple platforms. You can write SQL PL code that can run on DB2 for Linux, UNIX, Windows, i5/OS, and z/OS which improves the portability of your code. In addition, IBM supplies Eclipse based development tools that allow you to code, test, deploy, tune and debug your Native SQL Procedure from your desktop. Most Mid-tier or

desktop developers can be brought up to speed in a few days. No need to retrain on mainframe tools. In fact, I downloaded the free version of Data Studio from IBM's web site, installed it and had my first, very simple, Native SQL Procedure coded and tested in less than 6 hours. Nice! Thank you IBM!

So how do Stored Procedures work? Let's take a look at how it all started.

DB2 External Stored Procedures are user written application programs, usually COBOL, that are compiled, linked and cataloged into a load library that is made available to a stored procedure WLM address space. The BIND of the DBRM (SQL Statements) is performed and an Administrator executes the CREATE PROCEDURE statement to define the procedure to DB2.

External Stored Procedures

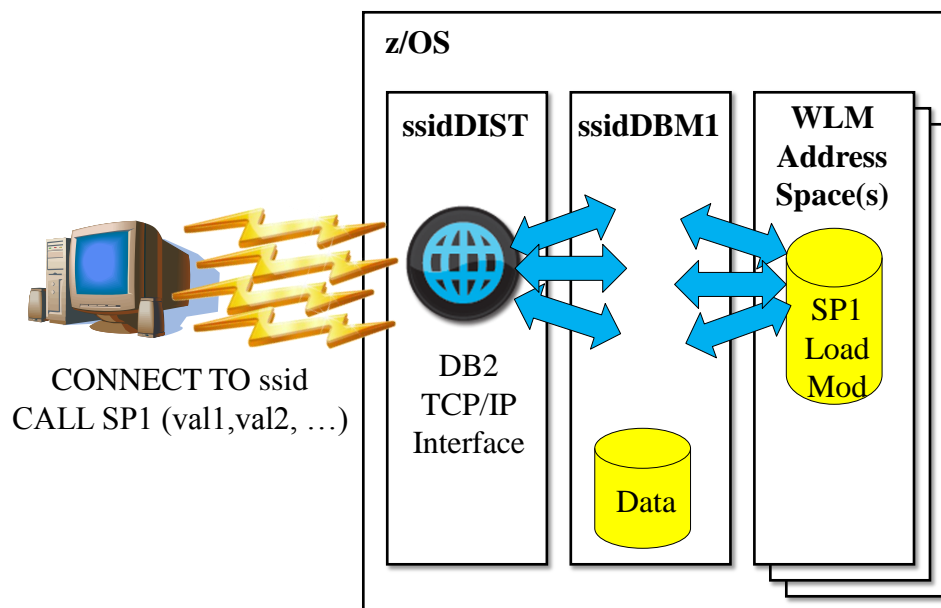


Figure 1. External Stored Procedure Environment

When a requesting application calls the Stored procedure from a local or remote location, as shown in Figure 1. the DB2 Data Base Services Address Space (DBM1) must look up the procedure in the SYSIBM.SYSROUTINES catalog table, acquires an available TCB to be used by the stored procedure and the stored procedure address space is notified to execute the stored procedure. If the procedure is not resident in the WLM address space it will be loaded from the load library. Control is then passed to the stored procedure, the stored procedure executes and

when an SQL statement is encountered control is passed back to the DBM1 address space to execute the bound package for the data access, and the result is passed back to the Stored procedure address space. When the Stored Procedure eventually completes control is returned to the DBM1 address space and the final result is passed back to the Caller.

In Version 5 of DB2 IBM introduced SQL Procedures. Today we refer to these procedures as External SQL Procedures. External SQL Procedures removed the host language requirement. The Procedures could be coded in it's entirety with the CREATE PROCEDURE statement. No host language coding skill required. However, the CREATE statement, with the definition information, SQL statements and procedural code must be converted to a host language application. Once the External SQL procedure is coded you can use the workbench tool set or execute the DSNHSQL procedure to translate the CREATE statement into a C language program and perform normal External Stored Procedure preparation. External SQL C programs run in WLM address spaces just like any other External Stored Procedure. Example:

```
CREATE PROCEDURE MYRAISE
  ( IN P_EMPNO CHAR(6)
    , IN P_RAISEPCT DEC(6,2)
  )
LANGUAGE SQL
UPDATE EMP
SET SALARY = SALARY * (1 + P_RAISEPCT/100)
WHERE EMPNO = P_EMPNO;
```

DB2 V9 introduced a new kind of SQL Procedure, the *Native SQL Procedure*. Unlike the original External SQL Procedures, there is no need to translate the procedure into a host language. When the CREATE PROCEDURE statement is executed DB2 inserts the definition information into the DB2 Catalog,

With V9 new function mode, when you create a native SQL procedure, its procedural statements are converted to a native representation that is stored in the DB2 catalog and directory, as it is done with other SQL statements, it is bound into a DB2 package. The parameter list and procedure options are stored in the database catalog tables as in the prior releases. When you CALL a native SQL procedure, DB2 loads the native representation(Package) from the directory and the DB2 engine

executes the procedure. No WLM required! Everything is stored in the package, under DB2 Control.

Native SQL Procedure Execution

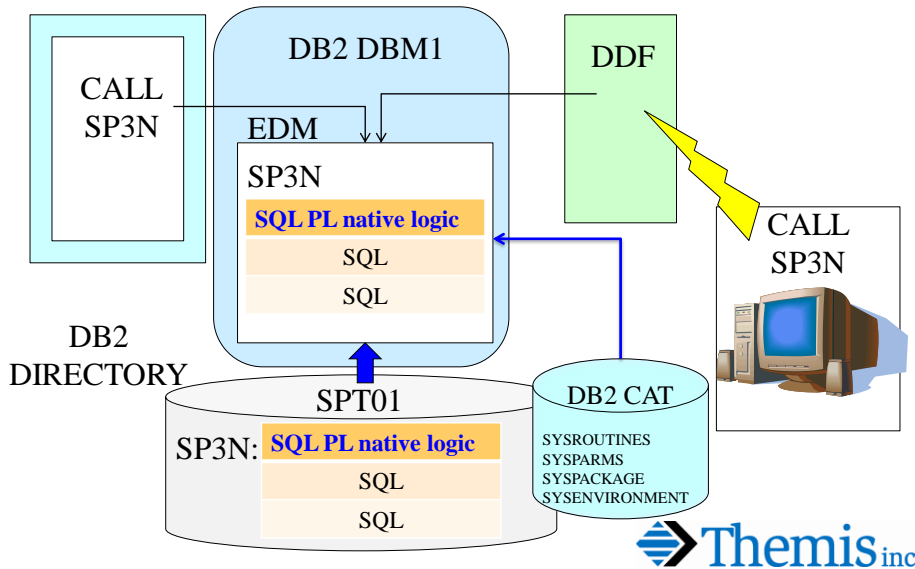


Figure 2. Native SQL Procedure Execution.

Native SQL Procedures are simply packages, with "runtime structures" for the SQL statements to be executed. So, when you invoke a native SQL procedure, DB2 finds and loads the package and executes the statements. Example:

```
CREATE PROCEDURE SPA80 (OUT p_CNT1 SMALLINT
                      ,OUT p_SUMSAL DECIMAL(11,2)
                      ,OUT p_SQLCODE INTEGER )
  VERSION V1
  ISOLATION LEVEL CS  VALIDATE BIND
  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
```

Performance

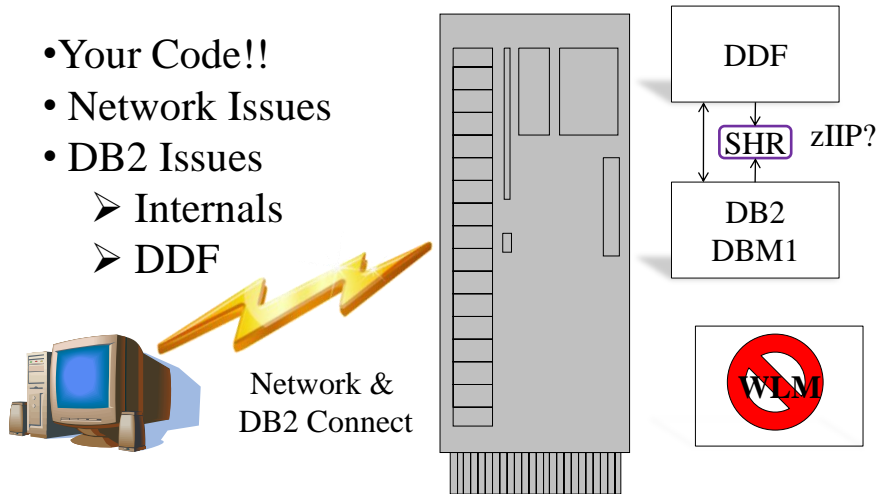


Figure 3. Native SQL Procedures and Performance

Unlike external stored procedures, native stored procedures do not execute in a WLM address space. WLM is only required for debugging.

With Native SQL procedure everything runs under the caller's task – when the stored procedure is called, the caller's DB2 thread just switches to the SQL procedures package; No queuing, no delays and reduced dispatch overhead.

Native SQL procedures are zIIP-eligible when called via DRDA.

Work that runs on the zIIP does not incur software charges based on the service units consumed; therefore it is a very attractive lower-cost alternative to running workloads on a general purpose processor. With the zIIP capability, the System z9 or System z10 mainframes help minimize the need to maintain duplicate copies of data. This eliminates the need to pass the data between DBM1 and the DDF address space. For server threads that process SQL requests from applications that access DB2 by TCP/IP, a portion of the SP executes under a dependent enclave SRB if it processes on behalf of an application that originated from an allied address space, or under an independent enclave SRB if the processing is performed on behalf of a remote application that accesses DB2 by TCP/IP. Shared memory is a relatively new type of virtual storage that allows multiple

address spaces to easily address common storage that is introduced in z/OS 1.5. It is similar to ECSA, since it is always addressable and no address register mode or cross memory moves are needed. It is different from ECSA since it is not available to all address spaces on the system. Only those that are registered with z/OS as being able to share this storage have visibility to this storage. Shared memory resides above the 2 GB bar.

As you can see, Native SQL Procedures are Exciting! They are simple to develop, easy to deploy, are cross platform compliant and on the z Server perform very well and are "Cost Effective"

Part 2. Experiences in Implementing Native SQL Procedures



Linda F. Claussen

A frequent speaker at conferences and user groups, Linda's 30+ years of experience spans the various roles of application developer, project manager, Data Base Administration and Systems Programmer. She has frequently worked with new DB2 sites performing the initial DB2 install, providing training and consulting to the Project Leaders, DBAs and Application staff responsible for the initial DB2 pilot project and conducting performance design reviews to insure successful project implementation. Linda is a past member of the IBM DB2 GOLD Consultants program and past IDUG conference chair. Linda currently works for Themis, Inc. as a DB2 for z/OS consultant and technical trainer and is a current member of the IBM Consultant Advisory Council.

She can be reached at:  <http://www.themisinc.com>
lclaussen@themisinc.com **US 1-800-756-3000**