UNIT 6: STORED PROCEDURES

OVERVIEW

- **UNDERSTAND THE CONCEPTS AND ADVANTAGES OF STORED PROCEDURES**
- **CODE A CLIENT PROGRAM**
- **CODE A STORED PROCEDURE**
- **SETUP THE SYSIBM.SYSPROCEDURES TABLE**

Figure: 6.1 Stored Procedures

STORED PROCEDURES: FLOW

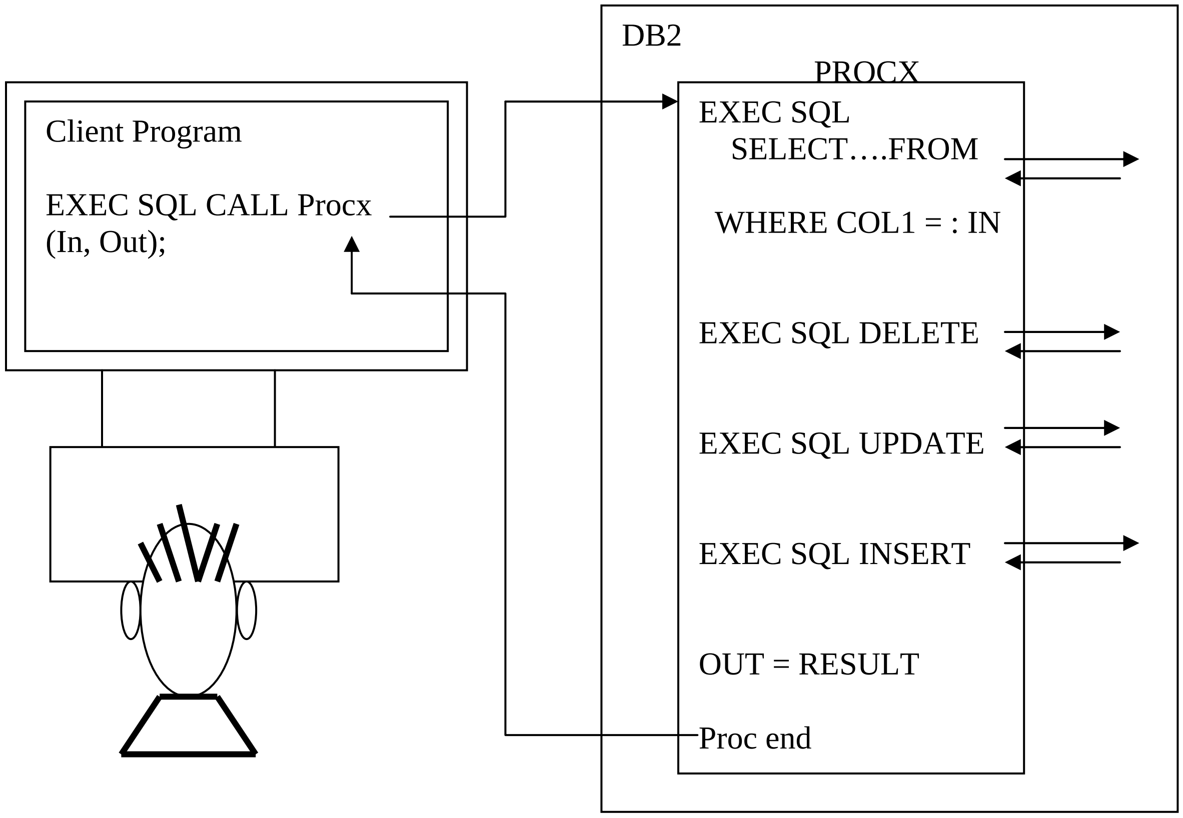


Figure 6.2 Stored Procedures: Flow

Notes :

A Stored procedure is a program that is controlled by DB2 that can be called through an SQL statement. The program can receive parameters and also has the ability to return data to the calling application.

STORED PROCEDURES: CONSIDERATIONS

- **RUN IN LE/370 ENVIRONMENT**
- **CODED IN ANY LANGUAGE(COBOL ,PL1, C)**
- **BEST TO CODE REENTRANT**
- **CAN NOT**
    - **ISSUE CAF CALLS**
    - **CALL OTHER STORED PROC'S**
    - **COMMIT/ROLLBACK**
- **CONNECT**
- **CAN USE NON-DB2 RESOURCE WITHOUT 2-PCS**
- **MUST BE DEFINED IN THE CATALOG**
- **CLIENT CAN BE LOCAL OR REMOTE**

Figure: 6.3 Stored Procedures Consideration

STORED PROCEDURES: CONSIDERATIONS (Cont...)

Both client and server applications can be written in any programming language. To make it a truly open concept, DB2 supports any type of client. The client can, in fact, be any application that can use either directly DB2 for MVS or that can pass calls to a DRDA application requester. The client and server are both shielded from any language differences by means of the LE/370 environment.

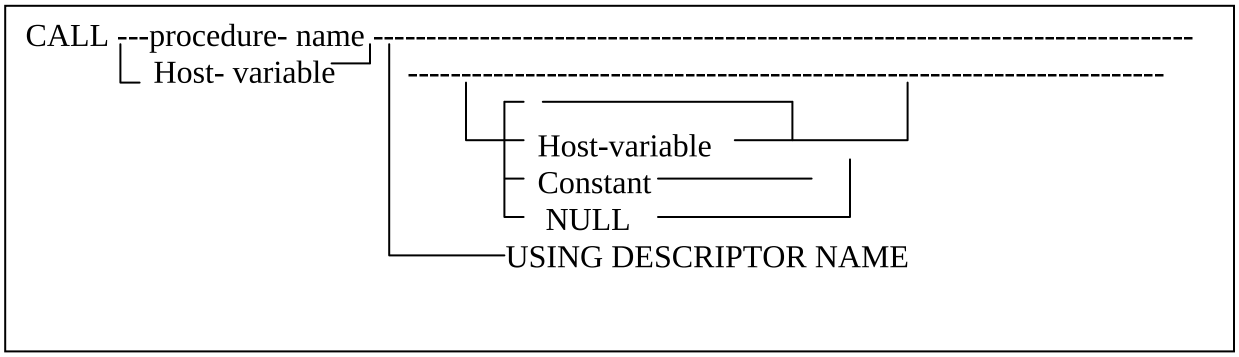
All SQL executed within the stored procedure will be considered to be within the same logical unit of work as the client application. DB2 will, therefore, coordinate the potential changes with the UOW of the calling application. This feature restricts the use of SQL-statements such as COMMIT and ROLLBACK. Since stored procedures run using the CAF (Call Attachment Facility), you will not be able to issue any CAF calls.

In the current implementation of DB2, there is only one address space that will run the procedures. It is, therefore, best to code the procedures as being reentrant, which will enable the parallel execution of the program. If it is not reentrant, only one user can use the procedure at a certain point in time.

The stored procedure could also access non-DB2 data such as CICS transactions.
That environment, however, is not going to be included in the same logical unit of work. There is no two-phase commit support with the other environments.
Interfaces the stored procedure can use are
The MQI interface (Message Queuing) for asynchronous execution, or the EXCI (External CICS interface) for synchronous execution of CICS transactions.

Figure: 6.4 Stored Procedures Consideration

STORED PROCEDURES: CALL SYNTAX



EXEC SQL CALL A (:EMP, :PRJ, :ACT)

EXEC SQL CALL A (:EMP :INDEMP, : PRJ :INDPRJ, :ACT :INDACT)

EXEC SQL CALL A (‘103455’, ‘BUIL5’, :ACT)

EXEC SQL CALL A USING DESCRIPTOR :SQLDA

EXEC SQL CALL : PROCNAM USING DESCRIPTOR : SQLDA

Figure: 6.5 Stored Procedures Call Syntax

Notes:

A number of different ways exist. Bottom line is that you can use all options as host variables, both the name of the procedure and the parameters that are passed to it.

STORED PROCEDURES : LINKAGE CONVENTIONS

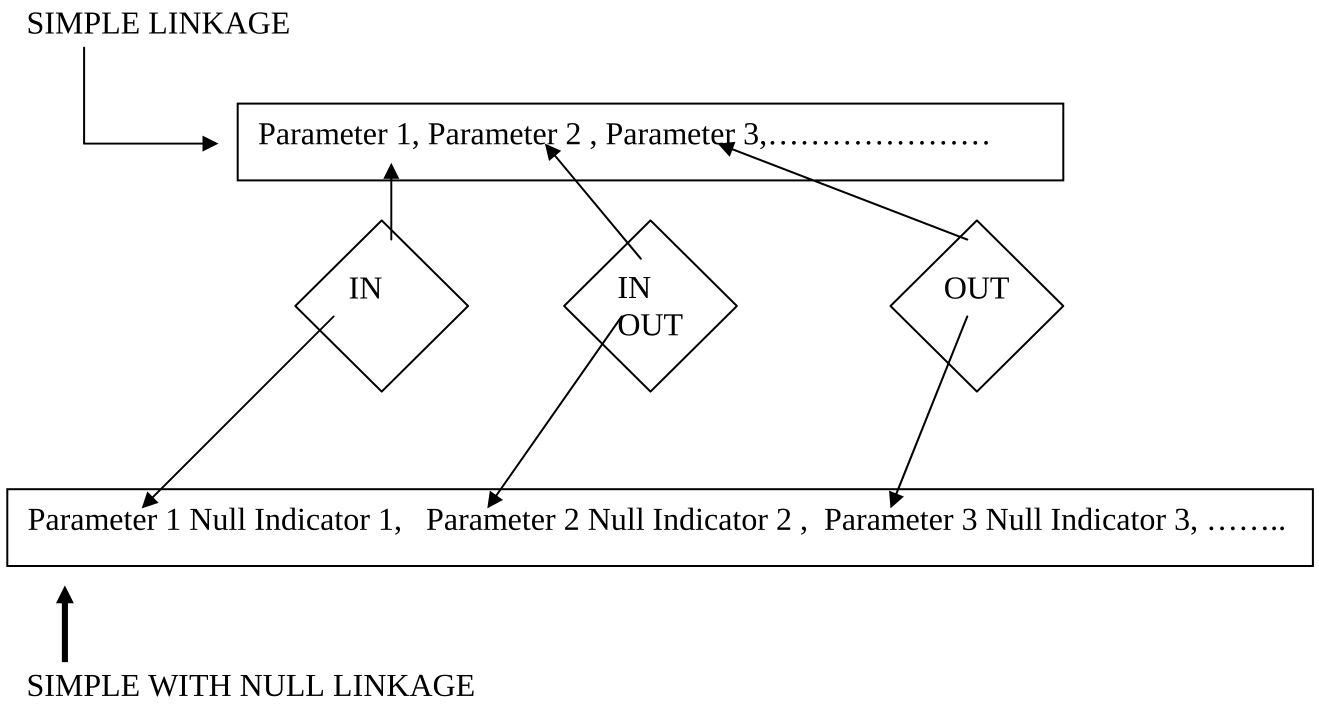

SIMPLE LINKAGE
Parameter 1, Parameter 2 , Parameter 3,…………………
IN
IN OUT
OUT
Parameter 1 Null Indicator 1, Parameter 2 Null Indicator 2 , Parameter 3 Null Indicator 3, ……..
SIMPLE WITH NULL LINKAGE

Figure: 6.6 Stored Procedure: linkage convention

Notes:

Two linkage conventions can be used:

SIMPLE :
This type of linkage only allows the use of null for parameters that are defined as output parameters. For the output parameters you can add an indicator Variable. If that variable is set to a negative value, DB2 will not pass the whole variable to the procedure. This technique will help improve the speed of the call, as DB2 has less information to send.

STORED PROCEDURES : LINKAGE CONVENTIONS (Cont..)

- **SIMPLE WITH NULL** : This type of linkage will require a null indicator for all parameters regardless of whether they are IN, OUT or INOUT.

For both types of linkages, you will have to define the nature of the parameter, which can be:

- **IN** (this parameter contains no value upon return from the procedure)
- **OUT** (the content is not passed along to the procedure)
- **INOUT** (data flows in both directions)

Figure: 6.7 Stored procedures: linkage conventions

STORED PROCEDURES : DB2 SETUP

PROCEDURE	**:**	**Name of the procedure**
AUTHID	**:**	**Userid**
LUNAME	**:**	**Luname this entry is intended for**
LOADMODULE	**:**	**Name of the program used for this procedure**
COLLID	**:**	**Collection in which the package is stored**
LINKAGE	**:**	**'SIMPLE' or 'SIMPLE WITH NULLS'**
LANGUAGE	**:**	**Programming language used to code the program**
RUNOPTS	**:**	**Any option you want to pass to the LE/370 environment.**
PARMLIST	**:**	**Complete description of the parameter list**
ASUTIME	**:**	**Maximum amount of CPU service units one run can consume**
STAYRESIDENT	**:**	**Program stays in storage after it has completed**
IBMREQ	**:**	**Is this an IBM or a User entry.**

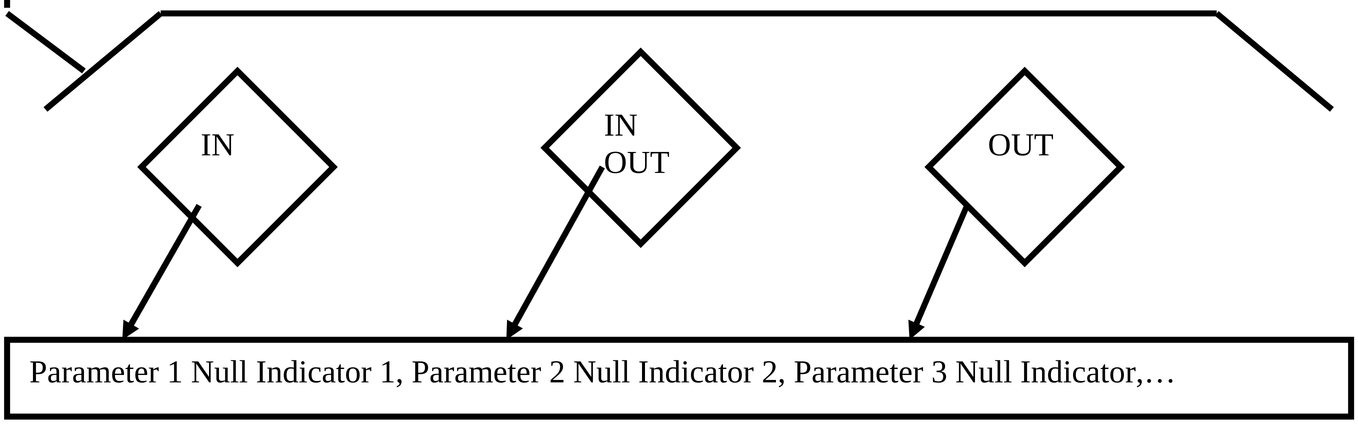

IN
IN OUT
OUT
Parameter 1 Null Indicator 1, Parameter 2 Null Indicator 2, Parameter 3 Null Indicator,…

Figure: 6. 8Stored Procedure : DB2 set up

STORED PROCDEURES: DB2 SETUP (Cont.)

To allow DB2 to know where and how the stored procedure is to be used, you will have to make an entry in a catalog table. This entry will give a description of who can use the procedure and what parameter conventions are used to call the procedure.

You can specify different definitions for the same procedure based on the calling LU-name and use rid. The will allow you to test a new procedure while the old one is still used.

The parm list has to contain a description of all the parameters that are included in the call and, of course, whether they are IN, OUT or INOUT.

The ASUTIME column allows you to put a limit on the amount of resources that can be consumed by one invocation of a stored procedure.

Figure: 6.9 Stored Procedures: DB2 setup (Cont.)