DB2 Information Management Software

Basic SQL Procedure Structure

Unit objectives

After completing this unit, you should be able to:

- Describe the structure of an SQL procedure
- Explain various clauses of the CREATE PROCEDURE statement
- List the statements that can be coded in the procedure body
- Alter Procedure
- Drop Procedure
- Create Module
- Replace Module
- Alter Module
- Drop Module

SQL stored procedures

- Based on ANSI/ISO standard language SQL/PSM
- Simple language which includes:
  - Features from block-structured languages
  - Exception handling
  - Familiar to Sybase, Oracle, Informix, Microsoft SQL Server programmers

SQL Procedure Language (1 of 3)

- SQL Procedures support:
  - Multiple parameters: input, output, input/output
  - Returning multiple output result sets to a client or to a calling SQL procedure
- SQL Procedures are defined in DB2 catalog
- SQL Procedure source is stored in DB2 catalog
- SQL Procedural Language (SQL PL) is folded to upper case
  - Exception: Delimited values

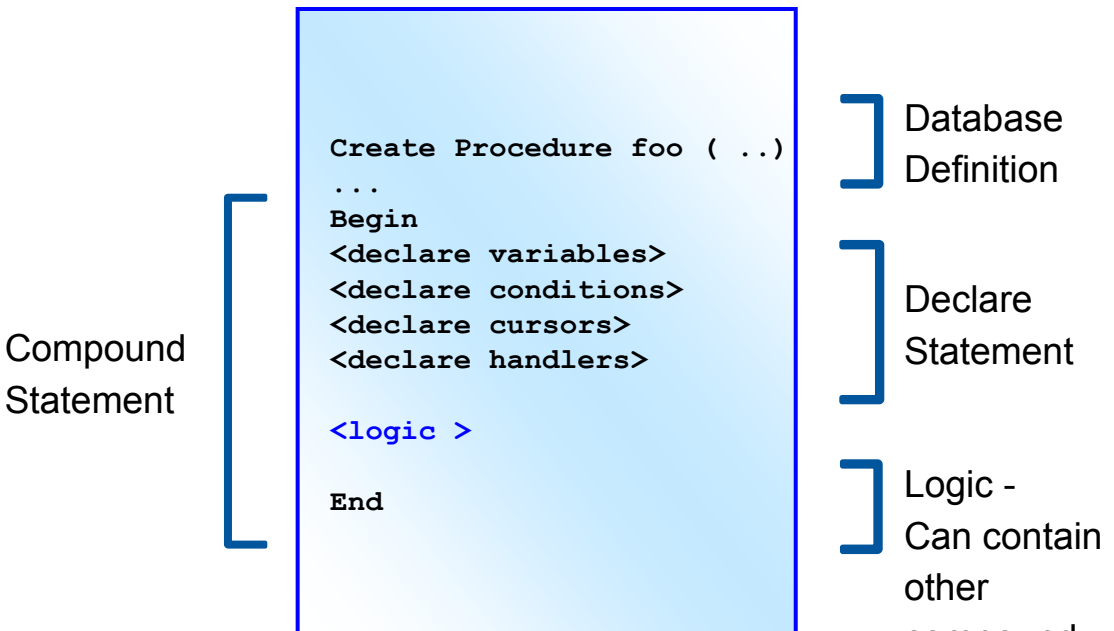
SQL Procedure Language (2 of 3)

```
1)  CREATE PROCEDURE DB2ADMIN.Sample1 ( IN in_Dept INT )
2)   RESULT SETS 1
3)   LANGUAGE SQL
-----------------------------------------------------------------
-- SQL Stored Procedure
-----------------------------------------------------------------
4)  P1: BEGIN
5)   DECLARE  r_error    int default  0;
6)   DECLARE  SQLCODE    int default 0;
7)   DECLARE CONTINUE HANDLER FOR SQLWARNING, SQLEXCEPTION, NOT
    FOUND
8)      BEGIN
    SET r_error = SQLCODE;
8)      END;
9)    BEGIN
a)    DECLARE cursor1 CURSOR WITH RETURN FOR
    SELECT   DEPTNAME,     MANAGER,  LOCATION
    FROM   ORG
    WHERE
      DEPTNUMB = in_Dept;
  -- Cursor left open for client application
b)    OPEN cursor1;
9)   END;
4)  END P1
```

SQL Procedure Language (3 of 3)

- **An SQL Procedure consists of:**
  - A CREATE PROCEDURE statement
    - LANGUAGE SQL
  - A procedure body which may include:
    - Compound statement(s): BEGIN … END
    - Declaration statements
    - Assignment statements
    - Conditional statements
    - Iterative control structure: LOOPs, and so forth
    - Exception Handling
    - CALL another stored procedure

Structure (1 of 2)



Structure (2 of 2)

- **An SQL Procedure can be:**

```
– A single statement

   CREATE PROCEDURE Sample1 (OUT Parm1 CHAR(10))
   LANGUAGE SQL
   SET Parm1 = 'value1'

–A compound statement

   CREATE PROCEDURE Sample2 (OUT Parm1 CHAR(10),
                             OUT Parm2 CHAR(10))
   LANGUAGE SQL
    BEGIN
      SET Parm1 = 'value1' ;
      SET Parm2 = 'value2';
    END

– Or nested compound statements
```

SQL Procedure Language statements

- Not limited to stored procedures
- Some platform differences
- Facilitate application solution
- Add business logic capability to SQL language

Where to use the "; "

```
CREATE PROCEDURE foo
    ( out day_Of_Year int )
LANGUAGE SQL
----------------------------------------
-- SQL Stored Procedure
----------------------------------------
P1: BEGIN
   DECLARE c_Date DATE;
   SET c_Date = CURRENT DATE;
  SET day_of_Year = dayofyear(c_Date);
END P1
```

Declarations (1 of 2)

- Local variables:

```
DECLARE var_name datatype [ DEFAULT value];
```

  - Example: DECLARE my_var INTEGER DEFAULT 6;

– Default value is NULL

– Variable name is folded to upper case

– Rules for ambiguous names:

  - First, check to see if there is an existing column of the same name (in one of the referenced tables)
  - When a column does not exist with that name, then check to see if there is an already defined SQL variable or parameter with the same name
  - Assumed to be a column name

Declarations (2 of 2)

- Condition declaration:

```
DECLARE not_found CONDITION FOR SQLSTATE '02000';
```

- Local cursor declaration:

```
DECLARE c1 CURSOR FOR select * from staff;
```

  - WITH RETURN TO CLIENT / WITH RETURN TO CALLER

- Handler declaration:

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION...;
```

Assignments

- **Syntax:**

```
SET lv_name = expression;
SET lv_name = NULL;
```

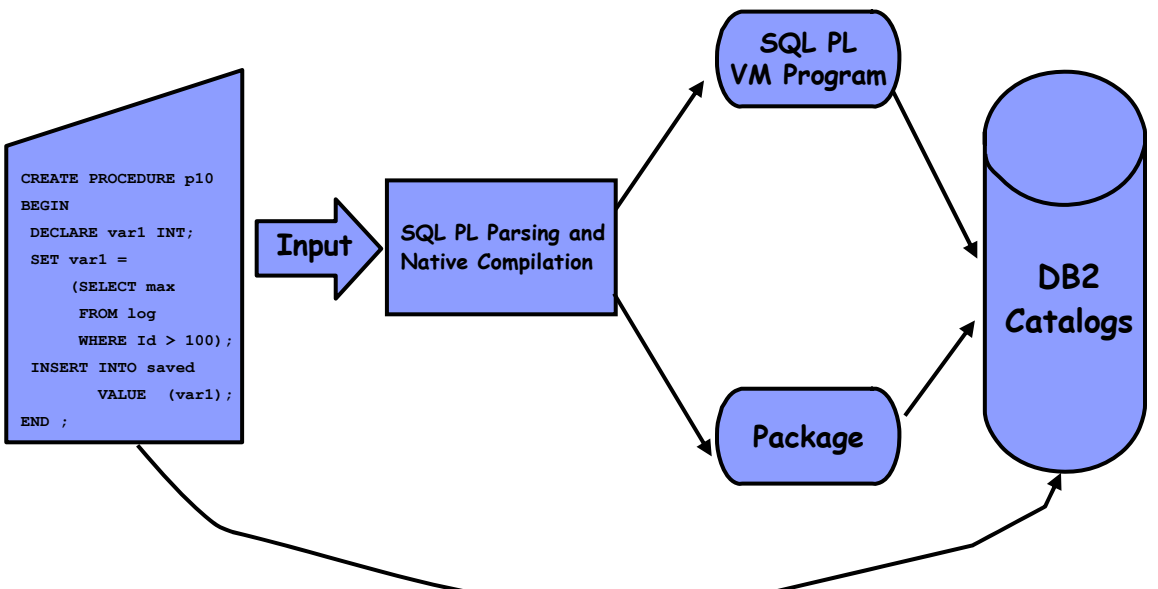
- **Example:**

```
SET salary = salary + salary*0.1;
SET init_salary = NULL;
SET salary = (select salary
              from employee
              where empno = lv_emp_num);
```

NOTE: An SQLERROR will occur if more than one row is returned when the SELECT statement in the preceding SET statement is executed.

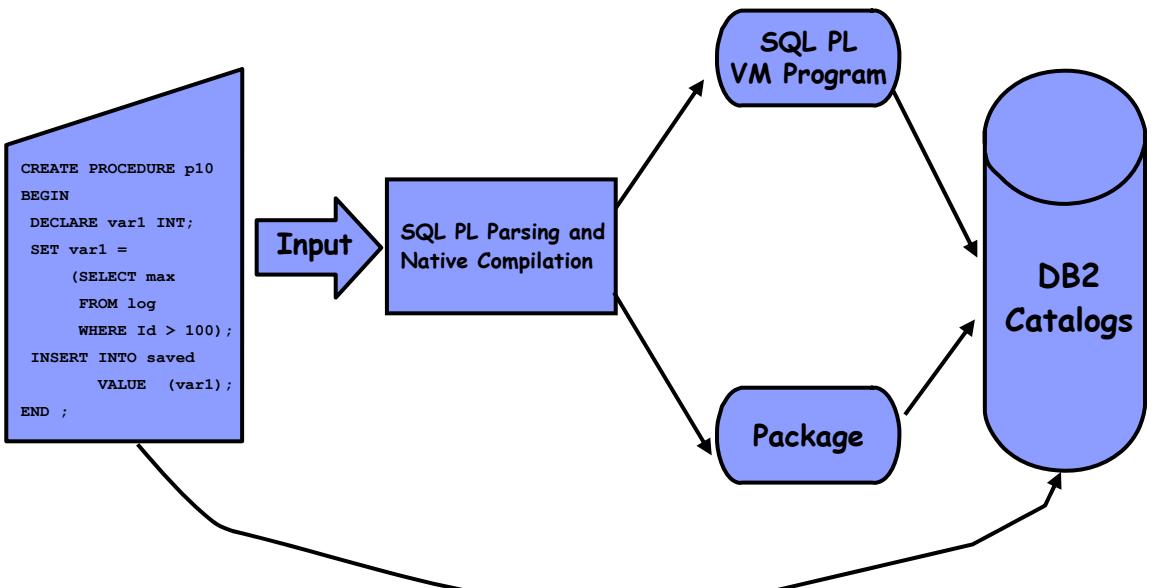
SQL procedures: Under the covers (1 of 2)

- Preparing an SQL procedure for execution



SQL procedures: Under the covers (2 of 2)

- How things work in DB2 for Linux, UNIX and Windows



Modules: Overview

- Module = bundle of several related objects:
  - SPs, UDFs, global variables and cursors, types, conditions
  - Similar to a class in OO languages (but single instance)

- Four main benefits:
  - Code organization/structure
  - Scoping
    - CALL mySchema.myModule.myProc()
  - Information hiding
    - Each object can be “public” or “private”
  - Global privilege control
    - Instead of granting/revoking on each SP, UDF or variable

Modules: Module specification

- Module that *exports* a type, a Stored Procedure, and a User-Defined Function

```
CREATE OR REPLACE MODULE myMod;

ALTER MODULE myMod PUBLISH
    TYPE myRowTyp AS ANCHOR ROW myTab;

ALTER MODULE myMod PUBLISH
   FUNCTION myFunc(val1 ANCHOR myTab.col1)
        RETURNS myRowTyp;

ALTER MODULE myMod PUBLISH
   PROCEDURE myProc(OUT parm1 ANCHOR myTab.col2);
```

Modules: Module implementation

```
ALTER MODULE myMod ADD
VARIABLE pkgVar ANCHOR myTab.col1;

ALTER MODULE myMod ADD FUNCTION
  myFunc(val1 ANCHOR myTab.col1)
  RETURNS myRowTyp
BEGIN
   DECLARE var1 myRowTyp;

   SELECT * INTO var1
       FROM myTab
       WHERE col1 < val1 AND col1
  > pkgVar;

   RETURN var1;
END
```

```
ALTER MODULE myMod ADD PROCEDURE
    myProc(OUT parm1 ANCHOR myTab.col2)
BEGIN
    DECLARE varRow myRowTyp;

    SET parm1 = varRow.col2 - pkgVar;
 END
```

Modules: Other statements

- DROP MODULE myMod;
  - Drops entire module
- ALTER MODULE myMod DROP BODY;
  - Drop “implementation”, keeps “specification”
- ALTER MODULE myMod DROP PROCEDURE myProc;
  - Drops module object
- GRANT EXECUTE ON MODULE myMod TO joe;
  - Grants user joe execute privilege on all routines and access to all variables and types in myMod

Checkpoint (1 of 2)

1. True or False? The stored procedure SQL Procedural Language is part of the DDL CREATE statement.

   _____

2. What would you find in a DECLARE statement within a stored procedure?

   _____

   _____

   _____

3. How is an assignment made to a parameter or a local variable?

   _____

   _____

   _____

Checkpoint (2 of 2)

4. True or False? GRANT EXECUTE ON MODULE myMod TO PUBLIC, does not grant execute privilege on all routines and access to all variables and types in myMod to all authorized users of the DB2 instance.

_____

Unit summary

Having completed this unit, you should be able to:

- Describe the structure of an SQL procedure
- Explain various clauses of the CREATE PROCEDURE statement
- List the statements that can be coded in the procedure body
- Alter Procedure
- Drop Procedure
- Create Module
- Replace Module
- Alter Module
- Drop Module