

DB2 Version 8 Inline SQL PL, Triggers and UDFs

DB2 Quickstart Education

Maintained by Paul Yip (ypaul@ca.ibm.com)

February 2003

IBM Software Group

Recommended Book



More info:

<http://www.software.ibm.com/data/developer/sqlplbook>

DB2 SQL Procedural Language for Linux, UNIX,
and Windows

ISBN: 0-13-100772-6

Presentation Notes

- This presentation assumes that you already know SQL PL and want to understand how it can be used in Triggers and UDFs. If not, please see DB2v8_SQLProcedures presentation first
- The Quicklabs in this presentation require that the SAMPLE database has been created
- Some examples are taken from the book: DB2 SQL Procedural Language for Linux, Unix and Windows (IBM Press)

Inline SQL PL

- SQL PL = SQL Procedural Language
- **Inline** SQL PL = SQL PL **subset** supported within a dynamic SQL statement, triggers and UDFs
- C compiler not required, logic executed inline with SQL in engine

Inline SQL PL - Supported Keywords

■ SUPPORTED:

SET
CASE
FOR
GET DIAGNOSTICS
GOTO
IF
RETURN
SIGNAL
WHILE
ITERATE
LEAVE

Unsupported SQL PL keywords

■ NOT SUPPORTED

ALLOCATE CURSOR
ASSOCIATE LOCATORS
DECLARE <cursor>
DECLARE ...HANDLER
PREPARE
EXECUTE
EXECUTE IMMEDIATE
LOOP
REPEAT
RESIGNAL
CALL
COMMIT/ROLLBACK

stored procedures
cannot be called from
triggers or functions

SELECT INTO ... sort of supported

- SELECT .. INTO ..

- instead, use:

SET (var1, var2) = (SELECT col1, col2 FROM table1); or

SET (var1, var2) = (SELECT col1, col2 FROM table1 FETCH FIRST 1 ROWS ONLY);

QuickLab: Dynamic compound SQL

- Setup: CREATE TABLE T3 (c1 INT)

```
BEGIN ATOMIC
  DECLARE cnt INT DEFAULT 0;
  DECLARE sumevens INT DEFAULT 0;
  DECLARE err_msg VARCHAR(1000) DEFAULT '';
  WHILE (cnt < 100) DO
    IF mod(cnt,2) = 0 THEN
      SET sumevens = sumevens + cnt;
    END IF;
    SET cnt=cnt+1;
  END WHILE;
  INSERT INTO T3 values (sumevens);
```

--(continued on next page)

Quicklab (cont...)

--- (cont...)

```
SET cnt = (SELECT 0 FROM SYSIBM.SYSDUMMY1);
FOR curl AS SELECT * FROM T1 DO
    IF curl.c1 > 100 THEN
        SET cnt = cnt + 1;
    END IF;
END FOR;

SET err_msg = 'Rows with values > 100 is: ' ||
char(cnt);
SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT = err_msg;
END
```

DB2 Data Management Software

IBM

DB2 Data Management Software

IBM

business software

Triggers

IBM Software Group

Triggers

■ 3 Types:

- ▶ BEFORE
 - activation before row is inserted, updated or deleted
- ▶ AFTER
 - FOR EACH ROW
 - activate after each row is inserted, updated or deleted
 - FOR EACH STATEMENT
 - activate after all rows are inserted, updated, or delete by a single statement. **Activates even if no rows are affected.**
- ▶ INSTEAD OF
 - defined on VIEWS

BEFORE trigger

- Used to validate or pre-process data before inserting
- FOR EACH ROW support only
- INSERT, UPDATE & DELETE not supported

define
qualifier for
new values

```
CREATE TRIGGER default_class_end
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
WHEN (n.ending IS NULL)
SET n.ending = n.starting + 1 HOUR
```

optional WHEN

if no value
provided on
insert, column
is null

Quicklab: BEFORE trigger, using SQL PL

```
CREATE TRIGGER validate_sched
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
BEGIN ATOMIC
  -- supply default value for ending time if null
  IF (n.ending IS NULL) THEN
    SET n.ending = n.starting + 1 HOUR;
  END IF;

  -- ensure that class does not end beyond 9pm
  IF (n.ending > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT='class ending time is beyond 9pm';
  ELSEIF (n.DAY=1 or n.DAY=7) THEN
    SIGNAL SQLSTATE '80001'
    SET MESSAGE_TEXT='class cannot be scheduled on a weekend';
  END IF;
END
```

atomic, dynamic
compound SQL

AFTER trigger

- Similar to BEFORE triggers, except that INSERT, UPDATE and DELETE are supported

```
CREATE TRIGGER audit_emp_sal
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
MODE DB2SQL
INSERT INTO audit VALUES (
  CURRENT_TIMESTAMP,
  ' Employee ' || o.empno || ' salary changed from ' ||
    CHAR(o.salary) || ' to ' || CHAR(n.salary) ||
  ' by ' || USER)
```

Final Notes

- Triggers, support a single SQL statement. However, by using dynamic compound SQL, multiple statements can form a single statement. [Another benefit of dynamic compound SQL is that inline SQL PL is supported](#)
- Where possible, use the WHEN clause to limit trigger activation
- For more information, see DB2 Developer Domain:
 - ▶ **Using SQL Procedural Language for Triggers in DB2 UDB**
<http://www7b.boulder.ibm.com/dmdd/library/techarticle/yip/0111yip.html>
 - ▶ **How to temporarily disable triggers in DB2**
<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0211yip/0211yip.html>
 - ▶ **Advanced SQL Procedural Scripting**
<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0203yip/0203yip.html>

User-defined Functions

Functions

- 4 Types:

- ▶ **Column Function**

- Example: SUM(), AVG()
 - not user-defineable

- ▶ **Scalar Function**

- Example: COALESCE(), SUBSTR()

- ▶ **Row Function**

- complements object relational features of DB2
 - not user-defineable

- ▶ **Table Function**

- Example: SNAPSHOT_DYN_SQL(), MREADALL()

- Since only Scalar and Table functions can be user defined, only those are covered in this presentation

Scalar Function

- Scalar functions take input values and return a single value

- Example:

if your application uses Oracle's NVL() function widely, it may be beneficial to simply map NVL() to DB2's COALESCE() function

```
CREATE FUNCTION NVL (p_var1 VARCHAR(30), p_var2 VARCHAR(30))  
SPECIFIC NVLVARCHAR30  
RETURNS VARCHAR(30)  
RETURN COALESCE(p_var1, p_var2)
```

return type

return value
(can also use
dynamic compound
SQL)

function
SPECIFIC name
useful if
overloading

QuickLab: Complex Scalar Function

- Example:

```
CREATE FUNCTION deptname(p_empid VARCHAR(6))
RETURNS VARCHAR(30)
SPECIFIC deptname
BEGIN ATOMIC
    DECLARE v_department_name VARCHAR(30);
    DECLARE v_err VARCHAR(70);
    SET v_department_name = (
        SELECT d.deptname FROM department d, employee e
        WHERE e.workdept=d.deptno AND e.empno= p_empid);
    SET v_err = 'Error: employee ' || p_empid || ' was not found';
    IF v_department_name IS NULL THEN
        SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT=v_err;
    END IF;
    RETURN v_department_name;
END
```

VARCHAR parameter type is recommended over CHAR. VARCHAR accepts both CHAR and VARCHAR input

Table Function

- used in the FROM clause of an SQL statement. the TABLE() function must be applied and must be aliased.

```
SELECT * FROM TABLE(SNAPSHOT_APPL_INFO('SAMPLE',-1)) T
```

TABLE() function

table function

alias

QuickLab: Table Function

- A function which enumerates a set of employees of a department

```
CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
  (enum INT, empno VARCHAR(6),
   lastname VARCHAR(15),
   firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
SELECT ROW_NUMBER() OVER(), e.empno, e.lastname, e.firstnme
FROM employee e
WHERE e.workdept=p_dept
```

← defines the table structure