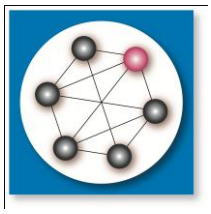

www.DBTechNet.org

SQL Stored Routine Labs

Workshop at Reutlingen University 2014-11-04
Martti Laiho.

Procedure Lab

procLab 1: Overloading

Experiment with the following polymorphism / overloading tests run in DB2 (pages 20-21 in “SQL Stored Routines” tutorial). Check if these apply to some other DBMS.

```
-- DB2:

-- Overloading on number of parameters:
db2 -td/
CONNECT TO testdb /
CREATE FUNCTION F (p1 INT)
RETURNS VARCHAR(30)
SPECIFIC myF1
RETURN 'p1'
/
CREATE FUNCTION F (p1 INT, IN p2 INT)
RETURNS VARCHAR(30)
SPECIFIC myF2
RETURN 'p1, p2'
/
CREATE FUNCTION F (p1 INT, p2 INT, p3 INT)
RETURNS VARCHAR(30)
SPECIFIC myF3
RETURN 'p1, p2, p3'
/
-- and test the overloading as follows:
CREATE TABLE DUAL (X CHAR(1) NOT NULL PRIMARY KEY, CHECK (X IN 'X')) /
INSERT INTO DUAL VALUES ('X') /
COMMIT
/
SELECT F(1,2) FROM DUAL /
SELECT F(1,2,3) FROM DUAL /
DROP SPECIFIC FUNCTION myF1 /
DROP SPECIFIC FUNCTION myF2 /
DROP SPECIFIC FUNCTION myF3 /

-- Overloading on data types
CREATE FUNCTION pmf (p1 INT)
RETURNS VARCHAR(10)
```

```
SPECIFIC myPMF1
RETURN 'p1 INT'
/
CREATE FUNCTION pmf (p1 REAL)
RETURNS VARCHAR(10)
SPECIFIC myPMF2
RETURN 'p1 REAL'
/
-- and test the overloading as follows:
SELECT pmf (1) FROM DUAL /
SELECT pmf (1.1) FROM DUAL /
DROP SPECIFIC FUNCTION myPMF1 /
DROP SPECIFIC FUNCTION myPMF2 /
```

procLab 2: DETERMINISTIC

Create and test a deterministic procedure which calculates VAT tax value to given amount using the given VAT%
Note: amount and VAT% cannot be negative, VAT% is less than 30%. Raise exception in case of errors.

procLab 3: Authors and books

Create the following tables

```
CREATE TABLE Authors(
id    INT  NOT NULL PRIMARY KEY,
names VARCHAR(200)  NOT NULL
);
CREATE TABLE Books (
id     INT NOT NULL PRIMARY KEY,
title  VARCHAR(50),
author_id  INT  REFERENCES Authors
);
INSERT INTO Authors VALUES (1, 'Walker E');
INSERT INTO Books VALUES
(100, 'My Hundred Miles', 1),
(101, 'Hundred and 1 Miles', 1),
(102, 'Return to Milestone 1', 1);
COMMIT;
```

Note: to keep this simple, the tables are not normalized!
Don't bother with the names problemacity.

Create and test following procedure with the signature

```
procedure InsertBook (author_id INT,
                      names VARCHAR,
                      book_id INT,
                      title VARCHAR)
```

which inserts new row to Authors table if author_id does not match with any id key in Authors table, inserts a row to the Books table for the book_id and title. In case of the book is already listed in the table, then the procedure raises exception on 'Book already registered'.

UDF Lab

funLab 1: DETERMINISTIC

Create and test a deterministic function which calculates VAT tax value to given amount using the given VAT %
Note: amount and VAT% cannot be negative, VAT% is less than 30%. Raise exception in case of errors.

funLab 2: Book title

```
INSERT INTO Books VALUES
(100, 'My Hundred Miles', 1),
(101, 'Hundred and 1 Miles', 1),
(102, 'Return to Milestone 1', 1);
COMMIT;
```

Consider the DB2 SQL PL function booktitle
(in "Getting Started With DB2 Application Development"
ebook of R. Chong et al)

```
CREATE FUNCTION booktitle(p_bid INTEGER)
RETURNS VARCHAR(300)
-------------------------------------------------------------
SQL UDF (Scalar)
-------------------------------------------------------------
SPECIFIC booktitle
F1: BEGIN ATOMIC
   DECLARE v_book_title VARCHAR(300);
   DECLARE v_err VARCHAR(70);
   SET v_book_title = (SELECT title FROM books WHERE p_bid = book_id);
   SET v_err = 'Error: The book with ID ' || CHAR(p_bid) ||
            ' was not found.';
   IF v_book_title IS NULL THEN
     SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT=v_err;
   END IF;
   RETURN v_book_title;
END F1
```

Apply and test this using Oracle PL/SQL and MySQL SQL/PSM languages.

Trigger lab

See the Parent-Child table pair

```
DROP TABLE Child;
DROP TABLE Parent;
CREATE TABLE Parent (
pid    INT NOT NULL PRIMARY KEY,
ps     VARCHAR(20),
psum   INT DEFAULT 0
);
CREATE TABLE Child (
cid    INT NOT NULL,
pid    INT,
cs     VARCHAR(20),
csum   INT,
CONSTRAINT Child_PK         PRIMARY KEY (cid),
CONSTRAINT Child_Parent_FK  FOREIGN KEY (pid)
```

```
   REFERENCES Parent (pid)
);

db2
connect to testdb
DELETE FROM Child
DELETE FROM Parent
INSERT INTO Parent (pid, ps) VALUES (1,'pa1')
INSERT INTO Parent (pid, ps) VALUES (2,'pa2')
INSERT INTO Child (cid, pid, cs, csum) VALUES (1, 1,'kid1',10)
INSERT INTO Child (cid, pid, cs, csum) VALUES (2, 1,'kid2',20)
INSERT INTO Child (cid, pid, cs, csum) VALUES (3,NULL,'orp3', 30)
COMMIT
SELECT * FROM Parent
SELECT * FROM Child
```

trigLab 1: Replacing CHECK constraint

Create trigger(s) which require that csum on rows of Child table must be between 0 and 30.

trigLab 2: Tracing

Apply the myProc examples in the tutorial and write a trigger which writes tracing information on INSERT, UPDATE and DELETE operations to the Child table. Test the trigger on Oracle or MySQL.

trigLab 3: RI INSERT rule

Consider removing the FOREIGN KEY of the Child table and implementing the RI INSERT rule (only those pid values are allowed which can be found as primary keys in the Parent table) for the Child table using applicable triggers.

---

trigLab 4: Controlling limit on the member rows

Design and test trigger(s) to control that only maximum number of members (maxm) can belong to the same team in the Team-Member table pair listed below adapted for DB2 CLP sessions.

a) Apply the tables, trigger(s) and test contents to DB2, or MySQL.
b) On Oracle you may have a problem. Why?

```
# Create the tables and triggers using terminal character /
db2 -td/
connect to testdb /
DROP TABLE Member /
DROP TABLE Team /

CREATE TABLE Team (
tid    INT NOT NULL PRIMARY KEY,
maxm   INT DEFAULT 0
) /

CREATE TABLE Member (
mid    INT NOT NULL,
tid    INT,
CONSTRAINT Member_PK         PRIMARY KEY (mid),
CONSTRAINT Member_Team_FK  FOREIGN KEY (tid)
   REFERENCES Team (tid)
) /
```

```
CREATE TRIGGER ..
-- design here the trigger(s) which control that
-- only maxm number of members can belog to every
-- single team
/

connect reset /
quit /

# Test run in a new DB2 session using terminal character ;
db2 -t;
connect to testdb ;
DELETE FROM Member ;
DELETE FROM Team ;
INSERT INTO Team (tid, maxm) VALUES (1,1) ;
INSERT INTO Team (tid, maxm) VALUES (2,2) ;
INSERT INTO Member (mid, tid) VALUES (1, 1) ;
INSERT INTO Member (mid, tid) VALUES (2, 1) ;
INSERT INTO Member (mid, tid) VALUES (3, 2) ;
INSERT INTO Member (mid, tid) VALUES (4, 2) ;
INSERT INTO Member (mid, tid) VALUES (5, 2) ;
INSERT INTO Member (mid, tid) VALUES (3,NULL) ;
COMMIT ;
SELECT * FROM Team ;
SELECT * FROM Member ;
connect reset ;
quit ;
```