Shranjeni podprogrami v SQL

- Shranjeni podprogrami: procedure in funkcije, ki jih pogosto potrebujemo
- Poimenovani SQL bloki, ki jih lahko kličemo s parametri
- Lahko spreminjajo podatke ali vračajo rezultate
- Funkcija: vrne natanko eno vrednost kot rezultat
- Procedura: vrača vrednost v izhodnih argumentih
- Omogočajo modularno in razširljivo pisanje programov
- Žal so implementacije pogosto sistemsko odvisne.

Shranjeni podprogrami v SQL

- Parametri (predvsem v procedurah)
  - vhodni (IN), izhodni(OUT)
  - vhodno-izhodni (IN OUT)
- Pogosto potrebna uporaba postopkovnih dodatkov (spremenljivke, kurzorji, ...)
  - ISO/ANSI: SQL/PSM (Persistant Stored Modules).
  - PostgreSQL: PL/pgSQL, Oracle: PL/SQL, Microsoft: T-SQL
  - MySQL, IBM DB2: najbližja standardu SQL/PSM
- Deklaracija in uporaba podprogramov

```
CREATE PROCEDURE
      Test (a IN VARCHAR(10)) AS ... ;

CALL ali EXECUTE Test('abcd');

DROP PROCEDURE Test;
```

Primer izračuna shranjenega atributa

- V tabelo jadralec dodamo število rezervacij za vsakega jadralca.

```
ALTER TABLE jadralec
      ADD stRez  INTEGER DEFAULT  0 NOT NULL;
```

- Kako izračunamo vrednost tega atributa?

```
UPDATE jadralec j
        SET stRez =
                (SELECT COUNT(*)
                 FROM rezervacija r
                 WHERE r.jid = j.jid);
```

Kdaj je vse to zares potrebno izračunati?

Primer procedure (Oracle)

- Inicializiraj število rezervacij na poljubno vrednost (parameter).

```
CREATE PROCEDURE JADR_REZ_INIT
( INIT IN INTEGER DEFAULT 0 ) AS
BEGIN
   UPDATE jadralec j
     SET stRez = INIT;
END;

CALL JADR_REZ_INIT(0);
```

Primer procedure (MariaDB/MySQL)

- Inicializiraj število rezervacij na poljubno vrednost (parameter).

```
DELIMITER //
CREATE PROCEDURE JADR_REZ_INIT
(IN INIT INTEGER)
BEGIN
    UPDATE jadralec j
      SET stRez = INIT;
END//
DELIMITER ;
CALL JADR_REZ_INIT(0);
```

Primer procedure (PostgreSQL)

- Pozna le funkcije; procedura je funkcija, ki vrača tip **VOID**
- Klic procedure (funkcije) v stavku SELECT

```
CREATE FUNCTION JADR_REZ_INIT
( INIT IN INTEGER DEFAULT 0 ) RETURNS VOID AS
$telo$
BEGIN
   UPDATE jadralec j
   SET stRez = INIT;
END;
$telo$ LANGUAGE plpgsql;
```

```
-- Klic:

SELECT JADR_REZ_INIT(0);
```

Primer procedure (Oracle)

- Izračunaj dejansko število rezervacij.

```
CREATE PROCEDURE JADR_REZ AS
BEGIN
  UPDATE jadralec j
    SET stRez =
          (SELECT COUNT(*)
           FROM rezervacija r
           WHERE r.jid = j.jid);
END;

CALL JADR_REZ();
```

Primer procedure (MariaDB/MySQL)

- Izračunaj dejansko število rezervacij.

```
DELIMITER //
CREATE PROCEDURE JADR_REZ()
BEGIN
  UPDATE jadralec j
    SET stRez =
          (SELECT COUNT(*)
           FROM rezervacija r
           WHERE r.jid = j.jid);
END//

CALL JADR_REZ();
```

Primer procedure (PostgreSQL)

- Izračunaj dejansko število rezervacij.

```
CREATE FUNCTION JADR_REZ()
    RETURNS VOID AS
$telo_funkcije$
BEGIN
    UPDATE jadralec j
        SET stRez =         (SELECT COUNT(*)
                             FROM rezervacija r
                             WHERE r.jid = j.jid);
END;
$telo_funkcije$ LANGUAGE plpgsql;
```

```
-- Klic:

SELECT JADR_REZ();
```

Primer procedure in funkcije (Oracle)

- Izračunaj dejansko število rezervacij.

```
CREATE FUNCTION JADR_REZ_FUNC
( JJID IN INTEGER) RETURN INTEGER AS
x INTEGER;  -- Lokalna spremenljivka
BEGIN
  SELECT COUNT(*) INTO x
  FROM rezervacija r
  WHERE r.jid = jjid;
  RETURN x;
END;
```

```
CREATE PROCEDURE JADR_REZ AS
BEGIN
  UPDATE jadralec j
     SET stRez = JADR_REZ_FUNC(j.jid);
END;
```

Primer procedure in funkcije (MariaDB/MySQL)

- Izračunaj dejansko število rezervacij.

```
DELIMITER //
CREATE FUNCTION JADR_REZ_FUNC
( JJID INTEGER) RETURNS INTEGER
BEGIN
  DECLARE x INTEGER;            -- Lokalna spremenljivka
  SELECT COUNT(*) INTO x
     FROM rezervacija r
     WHERE r.jid = jjid;
  RETURN x;
END//
```

```
CREATE PROCEDURE JADR_REZ()
BEGIN
  UPDATE jadralec j
     SET stRez = JADR_REZ_FUNC(j.jid);
END//
```

Primer procedure in funkcije (PostgreSQL)

```
CREATE FUNCTION JADR_REZ_FUNC
    (JJID IN INTEGER)  RETURNS INTEGER AS
$tf1$
DECLARE
  x INTEGER;  -- Lokalna spremenljivka
BEGIN
  SELECT COUNT(*) INTO x
  FROM rezervacija r
  WHERE r.jid = jjid;
  RETURN x;
END;
$tf1$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION JADR_REZ()
          RETURNS VOID AS
$tf2$
BEGIN
  UPDATE jadralec j
     SET stRez = JADR_REZ_FUNC(j.jid);
END;
$tf2$ LANGUAGE plpgsql
```

Bazni prožilci (triggerji)

- Prožilec: sestavljen SQL stavek, podobne oblike kot shranjena procedura, vendar nima argumentov
- Izvede se avtomatsko kot stranski produkt spremembe neke poimenovane tabele
- Ne kličemo ga ročno, ampak ga sproži prožilni dogodek
- Uporaba:
  - preverjanje pravilnosti vnosev in integritetnih omejitev (tudi denormalizacija)
  - opozarjanje na potrebne uporabniške akcije ob spremembah
  - vzdrževanje seznamov sprememb v PB
- Stavčni in vrstični prožilci.

Sintaksa prožilcev dogodkov nad tabelami

- ISO standard:

```
CREATE TRIGGER
BEFORE | AFTER dogodek ON tabela
[REFERENCING sinonimi za stare ali nove vrednosti]
[FOR EACH ROW]
[WHEN (pogoj)]    -- pogoj za vrstico (kot WHERE)
BEGIN
   -- telo prožilca
END;
```

- Dogodki: INSERT, UPDATE, DELETE

Stavčni prožilci

- Stavčni prožilec se izvede le enkrat na stavek, ki spremeni tabelo
- Oracle: stavčni prožilci so privzeti.
- MariaDB/MySQL: ne podpirata stavčnih prožilcev (samo vrstične).
- PostgreSQL: stavčni prožilci so privzeti.

Primer stavčnega prožilca (Oracle)

```
CREATE TRIGGER IzracunajSteviloRezervacij
AFTER INSERT OR UPDATE OR DELETE ON rezervacija
-- za vsak stavek (ne MySQL)
BEGIN            -- PL/SQL blok
   UPDATE jadralec
      SET stRez =
           ( SELECT COUNT(*)
             FROM rezervacija
             WHERE rezervacija.jid = jadralec.jid)
END;
```

Primer stavčnega prožilca (PostgreSQL)

- V PostgreSQL prožilci nimajo definiranega telesa, ampak lahko le kličejo vnaprej definirane *prožilne funkcije*.
  - Prožilne funkcije nimajo argumentov in vračajo tip TRIGGER
  - V prožilnih funkcijah se ob klicu ustvarijo prožilne spremenljivke (NEW, OLD, ...)

```
CREATE FUNCTION JADR_REZ_TRIG()
   RETURNS TRIGGER AS ...

CREATE TRIGGER IzracunajSteviloRezervacij
AFTER INSERT OR UPDATE OR DELETE ON rezervacija
FOR EACH STATEMENT        --  Privzeto
EXECUTE PROCEDURE  JADR_REZ_TRIG();
```

Vrstični prožilci

- Vrstični prožilec se izvede za vsako spremenjeno vrstico
- Odvisno od vrste dogodka lahko referenciramo
  - stare vrednosti pred spremembo (OLD): DELETE, UPDATE
  - nove vrednosti po spremembi (NEW): INSERT, UPDATE
  - Oracle: v WHEN sklopu OLD in NEW uporabljamo normalno, znotraj BEGIN/END pa z dvopičjem :OLD, :NEW
  - Oracle: z REFERENCING sklopom lahko OLD in NEW preimenujemo
- Prednost vrstičnih prožilcev: izvedemo telo prožilcev samo za vrstice, ki so se zares spremenile
- Nerodno: pogosto moramo za vsako vrsto dogodka napisati svoj prožilec (zelo podoben ostalim).

Primer vrstičnega prožilca (INSERT)

```
CREATE TRIGGER IzracunajSteviloRezervacij_I
AFTER INSERT ON rezervacija
REFERENCING NEW AS nova            -- Alias za NEW
FOR EACH ROW    -- za vsako novo vrstico
BEGIN   -- PL/SQL blok
   UPDATE jadralec
   SET stRez = stRez +1
   WHERE jadralec.jid = :nova.jid;
END;
```

Primer vrstičnega prožilca (DELETE)

```
CREATE TRIGGER IzracunajSteviloRezervacij_D
AFTER DELETE ON rezervacija
REFERENCING OLD AS stara            -- Alias za OLD
FOR EACH ROW    -- za vsako novo vrstico
BEGIN   -- PL/SQL blok
   UPDATE jadralec
   SET stRez = stRez -1
   WHERE jadralec.jid = :stara.jid;
END;
```

Primer vrstičnega prožilca (UPDATE)

```
CREATE TRIGGER IzracunajSteviloRezervacij_U
AFTER UPDATE ON rezervacija
REFERENCING OLD AS stara NEW AS nova
FOR EACH ROW    -- za vsako novo vrstico
WHEN (stara.jid != nova.jid)
BEGIN   -- PL/SQL blok
    UPDATE jadralec
    SET stRez = stRez +1
    WHERE jadralec.jid = :nova.jid;
    UPDATE jadralec
    SET stRez = stRez -1
    WHERE jadralec.jid = :stara.jid;
END;
```

MySQL: shranjene procedure in prožilci

- Spremenimo ločilo za konec stavka (namesto podopičja):
  npr. DELIMITER //
- Razlike pri parametrih: IN, OUT, INOUT pred imenom
  npr. (IN INIT INTEGER) samo za procedure, funkcije imajo le IN argumente, ni privzetih vrednosti
- Deklaracija lokalnih spremenljivk znotraj BEGIN/END:
  npr. DECLARE x INTEGER;
- Ne uporablja AS, RETURNS namesto RETURN
- Samo en dogodek na prožilec (nima OR)
- Ni stavčnih prožilcev, ni aliasov za OLD in NEW
- Ne uporabljamo dvopičja: OLD namesto :OLD
- Ne pozna WHEN sklopa (lahko pa uporabimo proceduralni IF/END IF sklop)

MySQL: primer vrstičnega prožilca (INSERT)

```
DELIMITER //
CREATE TRIGGER IzracunajSteviloRezervacij_I
AFTER INSERT ON rezervacija
FOR EACH ROW    -- za vsako novo vrstico
BEGIN
   UPDATE jadralec
   SET stRez = stRez +1
   WHERE jadralec.jid = NEW.jid;
END//
```

MySQL: primer vrstičnega prožilca (DELETE)

```
DELIMITER //
CREATE TRIGGER IzracunajSteviloRezervacij_D
AFTER DELETE ON rezervacija
FOR EACH ROW    -- za vsako novo vrstico
BEGIN
   UPDATE jadralec
   SET stRez = stRez -1
   WHERE jadralec.jid = OLD.jid;
END//
```

MySQL: primer vrstičnega prožilca (UPDATE)

```
DELIMITER //
CREATE TRIGGER IzracunajSteviloRezervacij_U
AFTER UPDATE ON rezervacija
FOR EACH ROW    -- za vsako novo vrstico
BEGIN
    IF NEW.jid != OLD.jid THEN
        UPDATE jadralec
        SET stRez = stRez +1
        WHERE jadralec.jid = NEW.jid;
        UPDATE jadralec
        SET stRez = stRez -1
        WHERE jadralec.jid = OLD.jid;
    END IF;
END//
```