MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

Databázové systémy a SQL

Lekce 8

Daniel Klimeš

Uživatelské funkce

- Objekty databáze, stejně jako tabulky
- Vytvoření příkazem CREATE, zrušení příkazem DROP
- Možné sdílení mezi uživateli, lze definovat oprávnění na spuštění
- Skládá se z DML SQL příkazů
- Konstrukce jazyka **PL/pgSQL** – Procedural Language
- Návratová hodnota
- Použití SELECT funkce()

Základy PL/pgSQL

- Standardní procedurální programovací jazyk, obdoba C, Java, Python
- Příkazy se vykonávají postupně + programovací smyčky

Základní prvky

- Bloky kódu ohraničeny BEGIN END
- Definice proměnných
- Operátor přiřazení hodnoty do proměnné
- Podmíněný výraz
- Programovací smyčka
- Volání jiných funkcí
- Prvky odděleny středníkem

Kostra funkce

```
CREATE OR REPLACE FUNCTION nazev_funkce ([parametr datovy_typ,…])
RETURNS typ_navratove_hodnoty AS $$
[DECLARE
    Nazev_promenne datovy_typ; …]
BEGIN
    telo_funkce
    RETURN navratova_hodnota;
END;
$$ LANGUAGE PLPGSQL;
```

Smazání funkce

```
DROP FUNCTION nazev_funkce ([parametr datovy_typ,…]);
```

Použití funkce

```
SELECT nazev_funkce ([parametr datovy_typ,…]);
```

PL/PGSQL - FOR

```
DECLARE
   rs RECORD; --deklarace kurzoru
BEGIN
FOR rs IN (SELECT * FROM patients) LOOP
          IF (rs.date_of_birth > current_date) THEN
                    INSERT INTO test_tab (patient_id) VALUES (rs.patient_id);
          END IF; -- ukončení podmíněného výrazu
END LOOP; -- ukončení smyčky
END;
```

- FOR rs IN (SELECT * FROM patients) LOOP
  - Příkaz smyčky
  - Proměnná rs (kurzor, „vektor“) postupně nabývá hodnot řádků, které vrací SELECT příkaz (jednotlivé pacienty)
  - Kurzor rs se musí deklarovat jako RECORD
  - Pro každý vrácený řádek SELECT příkazu se provedou příkazy uzavřené mezi LOOP a END LOOP
  - Smyčka končí po zpracování všech záznamů SELECTU
  - Pokud SELECT nevrací žádné řádky, blok smyčky se přeskočí

PL/PGSQL - IF

```
BEGIN
FOR rs IN (SELECT * FROM patients) LOOP
        IF (rs.date_of_birth > current_date) THEN
                INSERT INTO test_tab (patient_id) VALUES (rs.patient_id);
        END IF; -- ukončení podmíněného výrazu
END LOOP; -- ukončení smyčky
END;
```

- IF (rs.date_of_birth > current_date) THEN
  - podmíněný výraz
  - pokud je splněna podmínka, provedou se příkazy mezi THEN a END IF
  - Pokud ne, pokračuje se až za END IF

Proměnná v PL/SQL

```
CREATE OR REPLACE FUNCTION testf()
RETURNS NUMERIC AS $$
DECLARE
   rs RECORD;
   i NUMERIC(3);
BEGIN
i:=0;
FOR rs IN (SELECT * FROM patients) LOOP
            IF (rs.date_of_birth > current_date) THEN
                        INSERT INTO test_tab (patient_id) values (rs.patient_id);
                        i:=i+1;
            END IF; -- ukončení podmíněného výrazu
END LOOP; -- ukončení smyčky
RETURN i;
END;
$$ LANGUAGE PLPGSQL;
```

- DECLARE – zahajuje blok definice proměnných, každá proměnná musí být deklarovaná na začátku kódu

Operátor přiřazení – :=

Výjimky

```
CREATE OR REPLACE FUNCTION testf()
RETURNS NUMERIC AS $$
DECLARE
   rs RECORD;
   i NUMERIC(3);
BEGIN
i:=0;
FOR rs IN (SELECT * FROM patients) LOOP
            IF (rs.date_of_birth > current_date) THEN
                        INSERT INTO test_tab (patient_id) values (rs.patient_id);
                        i:=i+1;
            END IF; -- ukončení podmíněného výrazu
END LOOP; -- ukončení smyčky
RETURN i;
EXCEPTION
   WHEN division_by_zero THEN - -konkrétní očekávaná chyba
      RAISE NOTICE 'Neumim delit nulou';
      RETURN i;
   WHEN OTHERS THEN  - - další chyby
       RAISE NOTICE 'Neco je spatne: %', SQLERRM;
       RETURN i;
END;
$$ LANGUAGE PLPGSQL;
```

Ověření data

```
SELECT datum, s, TO_CHAR(datum, 'FMdd.FMmm.yyyy') bez_nul,
TO_CHAR(datum, 'dd.mm.yyyy') plne, values FROM (
SELECT TO_DATE(SUBSTRING (values from '[0123]?\d\.[01]?\d\.\d{4}'),
'dd.mm.yyyy') datum, SUBSTRING (values from '[0123]?\d\.[01]?\d\.\d{4}') s,
values
FROM eav_string
WHERE values ~ '[0123]?\d\.[01]?\d\.\d{4}'
) a
WHERE s <> TO_CHAR(datum, 'FMdd.FMmm.yyyy') AND  s <>
TO_CHAR(datum, 'dd.mm.yyyy')
```

Ověření data

```
CREATE OR REPLACE FUNCTION is_date(sdatum varchar(30))
RETURNS NUMERIC AS $$
DECLARE
   i NUMERIC(1);
   datum DATE;
   sdatum_s_nulama VARCHAR(20);
   sdatum_bez_nul VARCHAR(20);
BEGIN
   i:=0;
   datum := TO_DATE(sdatum, 'dd.mm.yyyy');
   sdatum_bez_nul := TO_CHAR(datum, 'FMdd.FMmm.yyyy');
   sdatum_s_nulama := TO_CHAR(datum, 'dd.mm.yyyy');
   IF (sdatum = sdatum_bez_nul or sdatum = sdatum_s_nulama) THEN
            i:=1;
   ELSE
            i:=0;
   END IF;
RETURN i;
END;
$$ LANGUAGE PLPGSQL;

SELECT is_date('29.2.2016')
```

S ošetřenou výjimkou

```
CREATE OR REPLACE FUNCTION is_date(sdatum varchar(30))
RETURNS NUMERIC AS $$
DECLARE
   i NUMERIC(1);
   datum DATE;
   sdatum_s_nulama VARCHAR(20);
   sdatum_bez_nul VARCHAR(20);
BEGIN
   i:=0;
   datum := TO_DATE(sdatum, 'dd.mm.yyyy');
   sdatum_bez_nul := TO_CHAR(datum, 'FMdd.FMmm.yyyy');
   sdatum_s_nulama := TO_CHAR(datum, 'dd.mm.yyyy');
   IF (sdatum = sdatum_bez_nul or sdatum = sdatum_s_nulama) THEN
            i:=1;
   ELSE
            i:=0;
   END IF;
RETURN i;
EXCEPTION
   WHEN OTHERS THEN  -- chyby
      RETURN 0;
END;
$$ LANGUAGE PLPGSQL;
```

Použití funkce

```
SELECT * FROM eav_string WHERE is_date(values) = 1 limit 20
```

```
SELECT values, SUBSTRING (values from '[0123]?\d\.[01]?\d\.\d{4}')
FROM eav_string
WHERE is_date(SUBSTRING (values from '[0123]?\d\.[01]?\d\.\d{4}')) = 1
limit 20
```

Procedura - vytvoření časové osy

Přehled počtu zařazených pacientů po měsících:

```
CREATE VIEW mesicni_pocty AS
SELECT  TO_CHAR(date_of_enrollment, 'yyyy-mm') mesic, COUNT(*) pocet
FROM
patient_study WHERE study_id = 43
GROUP BY  TO_CHAR(date_of_enrollment, 'yyyy-mm')
ORDER BY 1
```

Chybí některé měsíce → **Vytvoření časové osy v pomocné tabulce**

- Tabulka KALENDAR, její naplnění funkcí PROC_KALENDAR
- CREATE TABLE kalendar (
  Mesic VARCHAR(10)
)

PL/PGSQL procedura

```
CREATE OR REPLACE FUNCTION proc_kalendar(od DATE, mesicu NUMERIC)
RETURNS NUMERIC AS $$
DECLARE
   i NUMERIC(3);
BEGIN
DELETE FROM kalendar;
  FOR i IN 0..mesicu-1 LOOP
    INSERT INTO kalendar (mesic) VALUES (to_char(od + (interval '1 month' * i), 'yyyy-mm'));
  END LOOP;
RETURN i;
EXCEPTION
   WHEN OTHERS THEN
       RAISE NOTICE 'Neco je spatne: %', SQLERRM;
       RETURN -1;
END;
$$ LANGUAGE PLPGSQL;
```

PL/SQL procedura

Spuštění:

SELECT proc_kalendar ('2010-01-01', 24);

Doplněný výpis:

```
SELECT k.mesic, COALESCE(mp.pocet,0) pocet FROM
kalendar k LEFT JOIN mesicni_pocty mp ON k.mesic = mp.mesic
ORDER BY k.mesic
```