

Databázové systémy a SQL

Lekce 8

Daniel Klimeš

- 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()

- 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

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**,...]);

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

```
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

```

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 deklarována na začátku kódu

Operátor přiřazení – :=

```

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;

```



```

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')

```

```

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;

```

```
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
```