MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

Databázové systémy a SQL

Lekce 9

Daniel Klimeš, Monika Kratochvílová

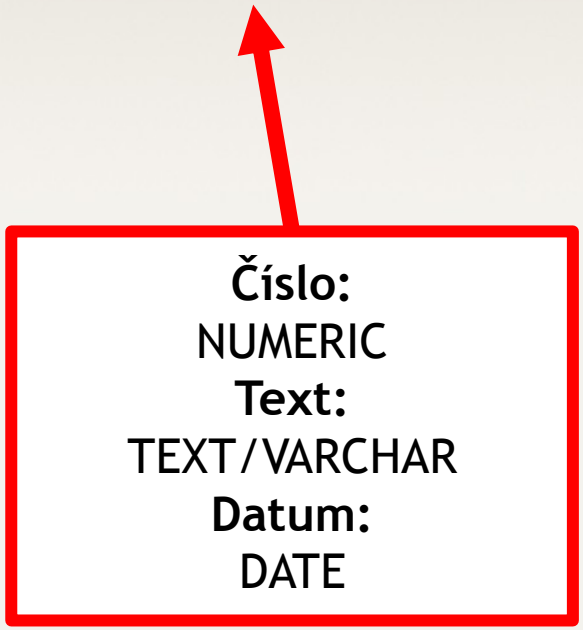
Funkce

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

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

Function

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



Do těla funkce se odkazuji na návratovou hodnotu pomocí dolaru a čísla určující pořadí argumentu (př.: $1)

Function

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

NUMERIC
TEXT/VARCHAR
TABLE
VOID

Funkce

Příklad: Funkce pro součet dvou hodnot.

```
CREATE OR REPLACE FUNCTION soucet (cislo1 NUMERIC, cislo2 NUMERIC)
RETURNS NUMERIC AS $$
BEGIN
RETURN $1 + $2; -- (nebo cislo1 + cislo2;)
END;
$$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION soucet2 (NUMERIC, NUMERIC)
RETURNS NUMERIC AS $$
DECLARE vysledek NUMERIC;
BEGIN
SELECT $1 + $2 INTO vysledek; -- (nebo lepe vysledek =  $1 + $2;)
RETURN vysledek;
END;
$$ LANGUAGE PLPGSQL;
```

Funkce - cvičení

Vytvořte funkci pro výpočet procenta.

Funkce - cvičení

Vytvořte funkci pro výpočet procenta.

```
CREATE OR REPLACE FUNCTION procento(numeric,numeric)
RETURNS NUMERIC AS $$
DECLARE
vysledek NUMERIC;
BEGIN
vysledek = ROUND(100*$1/$2, 2);
RETURN vysledek;
END;
$$ LANGUAGE PLPGSQL;
```

Funkce - cvičení

Vytvořte funkci pro výpočet věku pouze v letech.

Funkce - cvičení

Vytvořte funkci pro výpočet věku (pouze v letech).

```
CREATE OR REPLACE FUNCTION vek_roky(date)
RETURNS NUMERIC AS $$
DECLARE
vek NUMERIC;
BEGIN
vek = DATE_PART ('year', age($1));
RETURN vek;
END;
$$ LANGUAGE PLPGSQL;
```

Funkce - cvičení

Vytvořte tabulku vysetreni, která bude obsahovat id_pacienta, pocet_pred, pocet_po a datum_narozeni. A nainsertujte do tabulky dva libovolné záznamy.

Funkce - cvičení

Vytvořte tabulku vysetreni, která bude obsahovat id_pacienta, pocet_pred, pocet_po a datum_narozeni. A nainsertujte do tabulky dva libovolné záznamy.

```
SHOW DATESTYLE;
SET datestyle = "ISO, DMY";

CREATE TABLE vysetreni (
PACIENT_ID        NUMERIC(10),
POCET_PRED        NUMERIC(10),
POCET_PO          NUMERIC(10),
DAT_NAR           DATE
);

INSERT INTO vysetreni
VALUES (1,10,21,'23.02.1986'), (2,13,24,'25.12.1975');
```

Funkce - cvičení

Přidejte do tabulky vyšetření sloupec vek a naplňte jej pomocí vámi vytvořené funkce vek_roky.

Funkce - cvičení

Přidejte do tabulky vyšetření sloupec vek a naplňte jej pomocí vámi vytvořené funkce vek_roky.

```
ALTER TABLE vysetreni ADD COLUMN vek NUMERIC(2);

UPDATE vysetreni SET vek = vek_roky(datum_narozeni);
```

Funkce - cvičení

Přidejte do tabulky vyšetření sloupec změna_procento a naplňte jej pomocí vámi vytvořené funkce procento.

Funkce - cvičení

Přidejte do tabulky vyšetření sloupec zmena_procento, který bude znázorňovat procentuální změnu počtu před a po. A naplňte jej pomocí vámi vytvořené funkce procento.

```
ALTER TABLE vysetreni ADD COLUMN zmena_procento NUMERIC(4,2);
UPDATE vysetreni SET zmena_procento = procento(pocet_po - pocet_pred,
pocet_po);
```

-- poznámka
Do argumentu funkce lze vložit i SELECT:

```
ALTER TABLE vysetreni ADD COLUMN procento2 NUMERIC(4,2);
UPDATE vysetreni SET procento2 = procento(pocet_pred, (SELECT
sum(pocet_pred) FROM vysetreni));
```

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

1. Vytvoříte tabulku
2. Naimportujete data
3. Proveďte základní úpravu dat
4. Nakategorizujete některé proměnné
5. Všechny předchozí dotazy zaobalte do funkce

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

1. Vytvoříte tabulku

1. Stáhněte si ze sdíleného disku csv soubor mamo_scr.csv
2. Soubor otevřete v nějakém programu (např. excel)
3. Data si prohlédněte
4. Zjistěte, jaké sloupce soubor obsahuje a jakého jsou datového typu
5. Podle zdrojového souboru vytvořte tabulku v databázi

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

1. Vytvoříte tabulku

1. Stáhněte si ze sdíleného disku csv soubor mamo_scr.csv
2. Soubor otevřete v nějakém programu (např. excel)
3. Data si prohlédněte
4. Zjistěte, jaké sloupce soubor obsahuje a jakého jsou datového typu
5. Podle zdrojového souboru vytvořte tabulku v databázi

```
CREATE TABLE mamo_scr
(
id_pacientky        NUMERIC(10),
datum_narozeni      DATE,
kraj                VARCHAR(3),
metoda              NUMERIC(3),
datum_vys           DATE,
vysledek_vys        NUMERIC(3)
);
```

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

2. Naimportujete data

1. Napište COPY příkaz pro import dat z adresáře ve vašem počítači
2. Dejte si pozor na nastavení oprávnění – nastavte pro EVERYONE

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

2. Naimportujete data

1. Napište COPY příkaz pro import dat z adresáře ve vašem počítači
2. Dejte si pozor na nastavení oprávnění – nastavte pro EVERYONE

```
COPY mamo_scr
FROM
'C:document/…/mamo_scr.csv'
CSV delimiter ';' header null '';
```

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

3. Proveďte základní úpravu dat

1. Odstraňte všechny záznamy, kde chybí datum narození nebo datum vyšetření
2. Ověřte konzistenci datumů a případné chybné záznamy odstraňte
3. Doplňte hodnotu ve sloupci kraj na 999, kde kraj není uveden
4. Změňte hodnotu metody na NULL v případě, že nespadá mezi hodnoty 1,2,3,4,5,6,7,9 a hodnotu výsledku vyšetření v případě, že nespadá mezi hodnoty 1 až 10

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

3. Proveďte základní úpravu dat

1. Odstraňte všechny záznamy, kde chybí datum narození nebo datum vyšetření
2. Ověřte konzistenci datumů a případné chybné záznamy odstraňte
3. Doplňte hodnotu ve sloupci kraj na 999, kde kraj není uveden
4. Změňte hodnotu metody na NULL v případě, že nespadá mezi hodnoty 1,2,3,4,5,6,7,9 a hodnotu výsledku vyšetření v případě, že nespadá mezi hodnoty 1 až 10

```
DELETE FROM mamo_scr
WHERE datum_narozeni IS NULL OR datum_vys IS NULL OR vysledek_vys IS NULL;
DELETE FROM mamo_scr
WHERE datum_narozeni > datum_vys OR datum_vys > CURRENT_DATE
OR datum_narozeni > CURRENT_DATE;
UPDATE mamo_scr SET kraj = '999' WHERE kraj IS NULL;
UPDATE mamo_scr SET metoda = NULL WHERE metoda <1 OR metoda > 9 OR
metoda = 8;
UPDATE mamo_scr SET vysledek_vys = NULL WHERE vysledek_vys <1 OR
vysledek_vys > 10;
```

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

4. Nakategorizujete některé proměnné

1. Vytvořte nový sloupce vek (v letech) a vek_kategorie (0-44/45-69/70 a starší)
2. Vytvořte nový sloupec nador s hodnotami NEGATIVNI (vysledek_vys 1,2), BENIGNI (vysledek_vys 3,4), MALIGNI (vysledek_vys >=5) a NEZNAMO (vysledek_vys NULL)

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

4. Nakategorizujete některé proměnné

1. Vytvořte nový sloupce vek (v letech) a vek_kategorie (0-44/45-69/70 a starší)
2. Vytvořte nový sloupec nador (negativni/benigni/maligni/neznamo)

```
ALTER TABLE mamo_scr ADD COLUMN vek NUMERIC(2);
UPDATE mamo_scr SET vek=vek_roky(datum_narozeni);
ALTER TABLE mamo_scr ADD COLUMN vek_kategorie VARCHAR(20);
UPDATE mamo_scr SET vek_kategorie = CASE
        WHEN vek < 45 THEN '0 - 44'
        WHEN vek < 70 THEN '45 - 69'
        ELSE '70 a starší' END;
ALTER TABLE mamo_scr ADD COLUMN nador VARCHAR(20);
UPDATE mamo_scr SET nador = CASE
        WHEN vysledek_vys IN (1,2) THEN 'NEGATIVNI'
        WHEN vysledek_vys IN (3,4) THEN 'BENIGNI'
        WHEN vysledek_vys >= 5 THEN 'MALIGNI'
        ELSE 'NEZNAMO' END;
```

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

5. Všechny předchozí dotazy pro update a delete zaobalte do funkce

Funkce bez argumentu - cvičení

Vytvořte postupně funkci, v rámci které provedete:

5. Všechny předchozí dotazy pro update a delete zaobalte do funkce

```
CREATE OR REPLACE FUNCTION mamo_scr()

RETURNS VOID AS $$

BEGIN
Zde vložte všechny předchozí dotazy pro update a delete
(Pokud budete spouštět funkci opakovaně, musíte zajistit,
aby se vám data neduplikovala, proto musíte do funkce
vložit dotaz pro vymazání všech dat!!)

END;

$$ LANGUAGE PLPGSQL;


SELECT mamo_scr();
```

Funkce mamo_scr()

```
CREATE OR REPLACE FUNCTION mamo_scr()
RETURNS VOID AS $$
BEGIN
DELETE FROM mamo_scr;
COPY mamo_scr (id_pacientky, datum_narozeni, kraj, metoda, datum_vys, výsledek_vys)
FROM '/home/kratochvilova/vyuka/mamo_scr.csv, CSV delimiter ';' header null '';
DELETE FROM mamo_scr WHERE datum_narozeni IS NULL OR datum_vys IS NULL OR vysledek_vys IS NULL;
DELETE FROM mamo_scr WHERE datum_narozeni > datum_vys OR datum_vys > CURRENT_DATE OR
datum_narozeni > CURRENT_DATE;
UPDATE mamo_scr SET kraj = '999' WHERE kraj IS NULL;
UPDATE mamo_scr SET metoda = NULL WHERE metoda <1 OR metoda > 9;
UPDATE mamo_scr SET vysledek_vys = NULL WHERE vysledek_vys <1 OR vysledek_vys > 10;
UPDATE mamo_scr SET vek=vek_roky(datum_narozeni);
UPDATE mamo_scr SET vek_kategorie = CASE WHEN vek < 45 THEN '0 - 44, WHEN vek < 70 THEN '45 - 69,
ELSE '70 a starší' END;
UPDATE mamo_scr SET nador = CASE WHEN vysledek_vys IN (1,2) THEN 'NEGATIVNI,  WHEN vysledek_vys IN
(3,4) THEN 'BENIGNI, WHEN vysledek_vys >= 5 THEN 'MALIGNI, ELSE 'NEZNAMO' END;
END;
$$ LANGUAGE PLPGSQL;
```