MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

Databázové systémy a SQL

Lekce 7

Daniel Klimeš

Seskupení SQL příkazů

1. **SQL skripty**
2. **Procedury a funkce**

SQL skripty

•Skripty = seřazený seznam SQL DDL/DML příkazů
- CREATE, DROP, INSERT, UPDATE, DELETE
- •Příkazy odděleny středníkem
- Vytváření databázové struktury
- Jednorázové vkládání dat
- Transformace dat
- ORACLE
  - možnost tvořit jednoduché reportovací sestavy
  - příkaz SELECT
  - možnost použití proměnných
  - skript se spouští v **sqlplus** aplikaci
- FIREBIRD
  - •Skripty spustitelné stejně jako příkazy v IBConsole
  - •Nebo v aplikaci **isql**

Procedury a 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/SQL (ORACLE) – Procedural Language

Procedura	**x**	**Funkce**
•Vstupní a výstupní parametry • Lze spustit pouze v anonymním bloku nebo z jiné procedury		•1návratová hodnota • Použití v SELECT příkazu stejně jako např. funkce ROUND, SUBSTR, …

Základy PL/SQL

- Standardní procedurální programovací jazyk, obdoba C, Java, Pascal
- 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 procedur či funkcí
- Prvky odděleny středníkem

PL/SQL anonymní blok

- Ad-hoc spouštěný blok PL/SQL kódu
- Neukládá se, není součástí databáze
- Připomíná SQL skript
- Obsahuje PL/SQL konstrukce;
- Ohraničen BEGIN END

Příklad: **Id pacientů s chybným datem narození zapiš do pomocné tabulky**

```
BEGIN
    FOR rs IN (SELECT * FROM patients) LOOP
        IF (rs.date_of_birth > SYSDATE) THEN
            INSERT INTO test_tab (patient_id) values (rs.patient_id);
        END IF;
    END LOOP;
END;
```

PL/SQL - FOR

```
BEGIN – povinné otevření bloku
      FOR rs IN (SELECT * FROM patients) LOOP
            IF (rs.date_of_birth > SYSDATE) THEN
                  INSERT INTO test_tab (patient_id) values (rs.patient_id);
            END IF; -- ukončení podmíněného výrazu
      END LOOP; -- ukončení smyčky
END; – ukončení bloku
```

- FOR rs IN (SELECT * FROM patients) LOOP
  - příkaz smyčky
  - proměnná rs (kurzor, „vektor“) postupně nabývá hodnot řádků, které vrací SELECT příkaz (jednotlivé pacienty)
  - proměnná rs se nemusí deklarovat
  - 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/SQL - IF

```
BEGIN – povinné otevření bloku
      FOR rs IN (SELECT * FROM patients) LOOP
            IF (rs.date_of_birth > SYSDATE) THEN
                  INSERT INTO test_tab (patient_id) values (rs.patient_id);
            END IF; -- ukončení podmíněného výrazu
      END LOOP; -- ukončení smyčky
END; – ukončení bloku
```

- IF (rs.date_of_birth > SYSDATE) 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

```
DECLARE
i NUMBER;
BEGIN
i:=0;
DELETE FROM TEST_TAB;
FOR rs IN (SELECT * FROM patients) LOOP
   IF (rs.date_of_birth > SYSDATE) THEN
      INSERT INTO test_tab (patient_id) values (rs.patient_id);
      i:=i+1;
   END IF;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Celkem ' || i);
END;
```

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

Operátor přiřazení – :=

DBMS_OUTPUT.PUT_LINE('Celkem ' || i); - výpis ladící informace

Uložená procedura/funkce

```
CREATE OR REPLACE PROCEDURE jmeno_proc (parametry) IS
i NUMBER; -- deklarace proměných
BEGIN
    –tělo procedury
END;
```

```
CREATE OR REPLACE FUNCTION jmeno_funkce (parametry)
RETURN NUMBER IS
i NUMBER;
BEGIN
  --tělo funkce
END;
```

Parametry – (jmeno_parametru datovy_typ) odděleno čárkami
Např.: (datum DATE, cislo NUMBER)

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 procedurou PROC_KALENDAR

PL/SQL procedura

```
CREATE OR REPLACE PROCEDURE proc_kalendar (od DATE, mesicu
NUMBER) IS
i NUMBER;

BEGIN
  DELETE FROM kalendar;
  FOR i IN 0..mesicu-1 LOOP
    insert into kalendar (mesic) values (to_char(add_months(od, i), 'yyyy-mm'));
  END LOOP;

END proc_kalendar;
```

Spuštění:

```
begin
proc_kalendar(to_date('01.01.2010', 'dd.mm.yyyy'), 12);
end;
```

Doplněný výpis:

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

FIREBIRD - testovací databáze

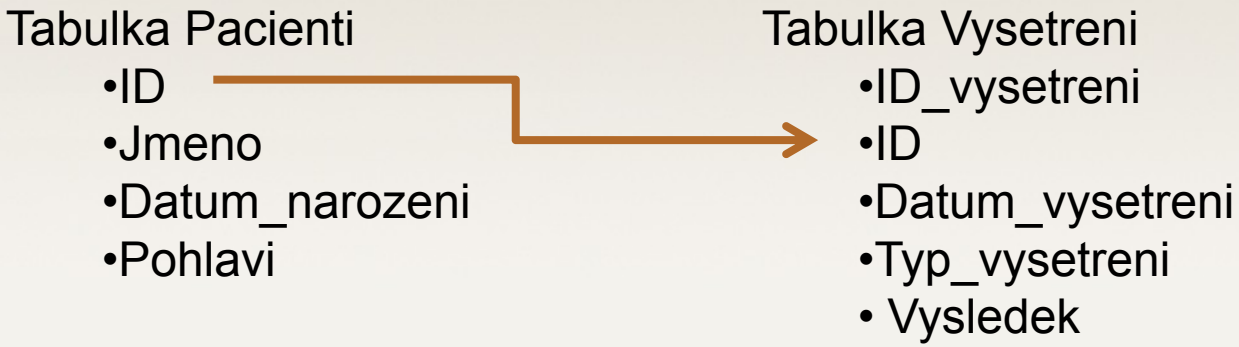
Tabulka Pacienti
- •ID
- •Jmeno
- •Datum_narozeni
- •Pohlavi

Tabulka Vysetreni
- •ID_vysetreni
- •ID
- •Datum_vysetreni
- •Typ_vysetreni
- • Vysledek

Minulé cvičení:
Zjistěte průměrný, minimální a maximální interval (počet dnů) mezi vyšetřeními pro jednotlivé typy vyšetření (vynechte vyšetření ve stejný den).
Jinak řečeno, jak často se má chodit na jednotlivá vyšetření.
Typ_vysetreni, průměr (dnů), min (dnů), max(dnů)

```
• SELECT v.typ_vysetreni,
•   AVG(v2.datum_vysetreni - v.datum_vysetreni),
•   MIN(v2.datum_vysetreni - v.datum_vysetreni),
•   MAX(v2.datum_vysetreni - v.datum_vysetreni)
•   FROM vysetreni v, vysetreni v2
•WHERE v.id = v2.id AND v.datum_vysetreni < v2.datum_vysetreni
•     AND v.typ_vysetreni = v2.typ_vysetreni
•     AND NOT EXISTS (
•       SELECT * FROM vysetreni v3 WHERE v3.id = v.id AND v3.typ_vysetreni = v.typ_vysetreni
•                               AND v3.datum_vysetreni > v.datum_vysetreni
•                               AND v3.datum_vysetreni < v2.datum_vysetreni
•       )
•GROUP BY v.typ_vysetreni
```

FIREBIRD - testovací databáze



Cvičení:

Vytvořte skript, který

1. Vytvoří tabulku „prvni_vysetreni“ –
   sloupce ID, Jmeno, Datum_narozeni, Pohlavi, Datum_vysetreni, typ vysetreni, vysledek
2. Přenese první vyšetření každého pacienta (podle datumu vyšetření) ze spojených tabulek Pacienti, Vysetreni do této nové tabulky
3. Smaže všechny muže
4. Změní výsledek 0 u všech záznamů na 1

•Příště – kolegyně Alena Zoláková – Jak přežít SQL