Kapitola 8

PL/SQL - Procedúra, Funkcia

8.1 Zadanie cvičenia - Procedúry, funkcie

1. Vytvorte procedúru `Vyskladaj_skupinu`, ktorá má 6 parametrov:
   - pracovisko - vstupný parameter - skratka pracoviska
   - odbor - vstupný parameter - číslo štúdijného odboru
   - zameranie - vstupný parameter - číslo zamerania
   - rocnik - vstupný parameter - číslo ročníka
   - kruzok - vstupný parameter - poradové číslo krúžku (1,2, . . . , 9, A, B, . . .)
   - st_skupina - **výstupný** parameter - vyskladaná skupina.

   Príklad:

   vstupy: Z, 100, 0, 1, 2 výstup: `5ZI012`
   Z, 101, 0, 3, A `5ZP03A`

Skratky odborov a zameraní je možné nájsť v tabuľke `priklad_db2.st_odbory`.	!

2. Predchádzajúcu procedúru prepíšte na funkciu `f_vyskladaj_skupinu`, kde pôvodný výstupný parameter bude výstupom funkcie.
3. Pokúste sa použiť funkciu `f_vyskladaj_skupinu` v selecte.
4. Vytvorte procedúru `Vloz_predmet`, ktorá vykoná INSERT operáciu nového predmetu do tabuľky `PREDMET`.
   Zabezpečte aby boli vyplnené všetky NOT NULL stĺpce ešte pred pokusom o vloženie.
   Skompilujte procedúru a vložte nasledujúce predmety:
   - Číslo predmetu: `BI14`, názov: `Databázové systémy`
   - Číslo predmetu: `BI12`, názov: `Úvod do inžinierstva`
   - Číslo predmetu: `BI12`, názov: `Úvod do inžinierstva`

   (Čo sa stalo pri vkladaní tohto záznamu? Nastala chyba? Prečo?)
5. Vytvorte príslušnú `exception` na ošetrenie predchádzajúcej chyby v procedúre `Vloz_predmet`.
6. Vytvorte funkciu `pocet_studentov` s parametrom predmet (cislo predmetu) a školský rok, ktorá vráti počet študentov, ktorí majú zapísaný daný predmet v danom školskom roku.

8.2 Domáca úloha - Procedúry a funkcie

1. Upravte procedúru `Vloz_predmet`, tak aby ste skontrolovali skôr ako to tam vložíte, či je možné vložiť a ak nie vypíšte hlášku, inak tam vložte nové údaje.
2. Prepíšte procedúru `Vloz_predmet` na funkciu, tak aby návratová hodnota bola:
   - `1` - ak riadok bol úspešne vložený,
   - `0` - ak sa nepodarilo riadok vložiť.
3. Vytvorte procedúru `Zmen_predmet`, ktorá vykoná UPDATE operáciu názvu predmetu v tabuľke `PREDMET`.
   - Ako vstupný parameter odovzdajte číslo predmetu a nový názov predmetu.
   - Vytvorte exception pre ošetrenie prípadu, keď zadaný predmet neexistuje, t.j. nenastane žiaden UPDATE.
   - Skompilujte procedúru a otestujte.
4. Vytvorte funkciu `Zrus_predmet`, ktorá vykoná DELETE operáciu predmetu v tabuľke `PREDMET`. Ako vstupný parameter odovzdajte číslo predmetu. Návratová hodnota funkcie bude počet vymazaných riadkov.
5. Vytvorte procedúru `Zapis_predmet`, ktorá zabezpečí zapísanie predmetu študentovi, pričom pred samotným vložením bude kontrolovať, či môže daný študent mať zapísaný daný predmet a správny počet kreditov.
   (a) Študent nemôže mať zapísaný predmet, ktorý už absolvoval.
   (b) Študent si môže zapísať predmet, ktorý absolvoval v predchádzajúcom stupni štúdia, len ak tento predmet je pre neho povinný, alebo povinne voliteľný v rámci tohto štúdia, pričom počet kreditov je nastavený na 0.

8.3 Jazyk PL/SQL

8.3.1 Deklarácia premenných

Deklarácie umožňujú v bloku uviesť zoznam definícií premenných, typov a kurzorov. Premenným môžeme nastaviť dátový typ, inicializovať ich na určitú hodnotu, dátový typ premennej nastaviť podľa tabuľky, prípadne stĺpca tabuľky.

```
Premenna         typ;                   --- premenná je daného typu
Premenna         typ := init_hodnota;   --- deklarácia a inicializácia
Premenna_record  tabulka%ROWTYPE;       --- záznam má položky rovnakého
                                        --- typu a názvu ako tabuľka
Premenna         tabulka.stlpec%TYPE;   --- premenná je rovnakého typu ako
                                        --- príslušný stĺpec tabuľky
```

8.3.2 Príkazy

8.3.3 Príkaz priradenia

```
Premenna := vyraz;
```

8.3.4 Prázdny príkaz

```
NULL;
```

8.3.5 Podmienková logika

Vetvenie je možné vykonať pomocou príkazov IF-THEN-ELSE, ELSIF a CASE.

8.3.6 Vetvenie príkazom IF

```
IF podmienka THEN
     Prikazy;
[ ELSE
     Prikazy;
END IF; ]
```

```
IF podmienka THEN
    Prikazy;
ELSIF podmienka2 THEN
    Prikazy;
[ ELSE
    Prikazy; ]
END IF;
```

```
IF podmienka THEN
    Prikazy;
ELSE IF podmienka2 THEN
             Prikazy;
    [ELSE
             Prikazy;]
    END IF;
END IF;
```

8.3.7 Vetvenie príkazom CASE

```
--od Oracle 9i
CASE premenna
   WHEN hodnota1 THEN prikazy1;
   WHEN hodnota2 THEN prikazy2;
   ...
   [ELSE prikazy; ]
END CASE;
```

```
--od Oracle 9i
CASE
   WHEN podmienka1 THEN prikazy1;
   WHEN podmienka2 THEN prikazy2;
   ...
   [ELSE prikazy; ]
END CASE;
```

- Ak v príkaze CASE neuvediete vetvu ELSE, PL/SQL pridá implicitnú ELSE klauzulu:

  ```
  ELSE RAISE CASE_NOT_FOUND;
  ```

- Príkaz CASE je možné použiť aj v SQL príkazoch, ale v tom prípade koniec príkazu je `END` a nie `END CASE`.

8.4 Cykly

Ak chceme spraviť opakovanie časti kódu, alebo spracovanie viacerých záznamov vrátených kurzorom, tak použijeme príkazy cyklu.

8.4.1 Základný cyklus LOOP

- Nekonečný cyklus:

```
LOOP
  Prikazy;
END LOOP;
```

- Ukončenie nekonečného cyklu pomocou EXIT:

```
LOOP
  ...
  IF podmienka THEN
    EXIT;
  END IF;
  ...
END LOOP;
```

```
LOOP
  ...
  EXIT WHEN podmienka;
  ...
END LOOP;
```

8.4.2 Cyklus WHILE

```
WHILE podmienka LOOP
  Prikazy;
END LOOP;
```

8.4.3 Cyklus FOR

```
FOR premenna IN min..max LOOP
  Prikazy;
END LOOP;
```

```
FOR premenna IN REVERSE min..max LOOP
    Prikazy;
END LOOP;
```

8.5 Nepomenovaný blok

```
[DECLARE              --- cast deklaracii
   nazov_premennej  typ[:= init_hodnota];
]
BEGIN
  Prikazy;           --- cast prikazov
[EXCEPTION            --- cast odchytenia a spracovania vynimiek
  WHEN  typ_vynimky THEN
    prikazy;
  WHEN typ_vynimky2 THEN
    prikazy;
  ...
]
END;
/
```

8.6 Procedúry a funkcie

Syntax procedúry

```
CREATE [OR REPLACE] PROCEDURE nazov_procedury
 [( parameter1 [ mode1] datatype1,
    parameter2 [ mode2] datatype2, . . .)]
IS|AS
  [ nazov_premennej  typ[:= init_hodnota]; ]
BEGIN
  Prikazy;
  [ EXCEPTION
      WHEN typ_vynimky THEN
         Prikazy;
      [WHEN ...]
  ]
END  [nazov_procedury];
/
```

Syntax funkcie

```
CREATE [OR REPLACE] FUNCTION nazov_funkcie
 [( parameter1 [ mode1] datatype1,
    parameter2 [ mode2] datatype2, . . .)]
RETURN datatype
IS|AS
  [ nazov_premennej  typ[:= init_hodnota]; ]
BEGIN
  Prikazy;
  RETURN vyraz;
[ EXCEPTION
      WHEN typ_vynimky THEN
         Prikazy;
      [WHEN ...]
  ]
END  [nazov_funkcie];
/
```

Mode - Typ argumentu:

- **IN** (default) – vstupný. Odovzdáva sa hodnota z volaného prostredia do procedúry ako konštanta. Pri pokuse o zmenu hodnoty argumentu, ktorý je definovaný ako IN, nastane chyba.

- **OUT** – výstupný. Odovzdáva sa hodnota argumentu do prostredia, odkiaľ bola procedúra volaná.

- **IN OUT** – vstupno-výstupný. Odovzdáva sa hodnota argumentu z prostredia a zmenená hodnota môže byť pomocou toho istého argumentu odovzdaná do prostredia, odkiaľ bola procedúra volaná.

Aby bolo možné procedúry vytvárať a používať, je potrebné zabezpečiť aj práva na tvorbu procedúr, funkcií (CREATE ANY PROCEDURE) a práva na vykonanie procedúr, alebo funkcií (EXECUTE).

```
GRANT CREATE ANY PROCEDURE TO uzivatel;
GRANT EXECUTE ON nazov_procedury TO uzivatel;
```

IN	OUT	IN OUT
default	Musí byť špecifikovaný	Musí byť špecifikovaný
Formálny parameter sa chová ako konštanta	Neinicializovaná premenná	Inicializovaná premenná
Parameter môže byť literál, výraz, konštanta alebo inicializovaná premenná	Musí byť premenná	Musí byť premenná

8.6.1 Príklady procedúr a funkcií

Príklad 8.1: Príklad procedúry

Majme procedúru `query_stud`, ktorá na základe vstupného parametra (osobné číslo), vráti celé meno študenta a jeho študijnú skupinu vo výstupných parametroch.

```
CREATE OR REPLACE PROCEDURE query_stud
  (v_oc       IN  student.os_cislo%TYPE,
   v_meno     OUT VARCHAR2,
   v_skupina  OUT student.st_skupina%TYPE)
IS
BEGIN
 SELECT meno||' '|| priezvisko, st_skupina
    INTO v_meno, v_skupina
  FROM student st JOIN  os_udaje ou USING (rod_cislo)
  WHERE st.os_cislo = v_oc;
END query_stud;
/
```

Príklad 8.2: Použitie procedúry v príkazovom riadku

```
-- deklarácia premenných v sqlplus
SQL> VARIABLE p_meno VARCHAR2(30)
SQL> VARIABLE p_skupina CHAR(5)

-- spustenie procedúry v sqlplus - pozor na :premenna
SQL> EXECUTE query_stud( 1512, :p_meno, :p_skupina);

PL/SQL procedure successfully completed.

-- vypis hodnoty premennej (bez : )
SQL> PRINT p_meno p_skupina

P_MENO
------------------
Peter Novak


P_SKUPINA
------------------
5Z012
```

Príklad 8.3: Použitie procedúry v nepomenovanom bloku

Aby bolo možné vypísať text na konzolu pomocou metódy `dbms_output.put_line`, je najprv nutné zadať nasledovný príkaz:

```
set serveroutput on
```

```
declare
  --- deklarácia premenných v bloku príkazov
  p_meno VARCHAR2(30);
  p_skupina CHAR(5);
begin
  --- spustenie procedúry v bloku
  query_stud( 1512, p_meno, p_skupina);

  --- vypis hodnoty premennej
  dbms_output.put_line ( 'Meno:     '|| p_meno);
  dbms_output.put_line ( 'Skupina : '|| p_skupina);
end;
/

 --- na konzole bude výpis:
  Meno:    Peter Novak
  Skupina: 5Z012
```

Príklad 8.4: Príklad funkcie

```
CREATE OR REPLACE FUNCTION pocet_kreditov (p_oc IN student.os_cislo%TYPE)
   RETURN NUMBER
IS
   p_kredity zap_predmety.ects%TYPE :=0;
BEGIN
  SELECT sum(nvl(ects,0)) INTO p_kredity
    FROM zap_predmety
    WHERE os_cislo = p_oc
      AND vysledok in ('A','B','C','D','E');

  RETURN p_kredity;

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN 0;
END pocet_kreditov;
/
```

Príklad 8.5: Použitie funkcie v príkazovom riadku

```
--- deklarácia premenných v sqlplus
SQL> VARIABLE kredity NUMBER
SQL> VARIABLE oc NUMBER
SQL> EXECUTE :oc := 500438

--- spustenie funkcie v sqlplus - pozor na :premenna
SQL> EXECUTE :kredity := pocet_kreditov(:oc)

--- výpis hodnoty - pozor bez :
SQL> PRINT kredity

   KREDITY
----------
       104
```

Príklad 8.6: Použitie funkcie v bloku príkazov

```
declare
```

```
  kredity number;
  oc   number := 500438;
begin
  kredity := pocet_kreditov(oc);      --- vykonaj funkciu
  dbms_output.put_line(kredity); --- vypis vysledok
end;
/
```

Príklad 8.7: Použitie funkcie v príkaze Select

```
SQL> SELECT st.os_cislo, ou.meno, ou.priezvisko, pocet_kreditov(st.os_cislo)
     FROM student st JOIN os_udaje ou USING (rod_cislo);
```

- Parametre v príkazovom riadku
  - Pri spustení metódy z príkazového riadku môžeme hodnotu vstupného parametra (IN) zadať priamo konštantou, alebo premennou.
  - Hodnotu vstupno-výstupného parametra (IN OUT) je nutné zadať pomocou premennej.
  - Pre získanie výstupnej hodnoty z výstupného parametra (OUT) procedúry je nutné zadať premennú.
  - Premennú je potrebné deklarovať pomocou príkazu VARIABLE (len názov premennej a dátový typ).
  - Pri použití premennej pri spustení metódy je nutné zadať pred meno znak ":"(priamo pred názov premennej).
  - Pri zisťovaní hodnoty (vstupno-)výstupnej premennej príkazom PRINT už sa nepoužíva znak ":"
- Funkcie
  - Rozdiel medzi funkciami a procedúrami je minimálny - funkcia svojim menom vracia hodnotu.
  - Funkcie je možné priamo využiť v príkazoch Select, pokiaľ návratová hodnota funkcie je typu, ktorý korešponduje s SQL typmi.(t.j. napr. nesmie vracať BOOLEAN hodnotu).
  - Návratovú hodnotu funkcie nie je možné "zahodiť", ale je nutné výsledok funkcie využiť v príkaze Select, alebo výsledok funkcie uložiť do premennej, alebo použiť ako vstup inej metódy, prípadne výrazu.

Upozornenie:
Názov parametra alebo premennej, ktoré chcete používať v príkaze Select **NESMIE** mať rovnaký názov ako stĺpec tabuľky (pohľadu), z ktorej vyberáme. Toto je závažná chyba, ktorú kompilátor nerozpozná, ale pravdepodobne bude viesť k nesprávnym výsledkom.

8.6.2 Spôsob odovzdávania parametrov

- **pozíciou** – premenné odovzdané procedúre v takom istom poradí ako sú deklarované.
- **názvom** – premenné odovzdané v ľubovoľnom poradí, každá hodnota je asociovaná s názvom premennej použitím syntaxe => .
- **kombinované** – prvé parametre odovzdané pozíciou, zbytok názvom.

Príklad 8.8: Spôsob odovzdávania parametrov

1. Majme procedúru s nasledovnou hlavičkou:

```
CREATE OR REPLACE PROCEDURE add_dept
( v_name  IN dept.dname%TYPE  DEFAULT 'unknown',
  v_loc   IN dept.loc%TYPE    DEFAULT 'unknown') ...
```

2. Potom možné spôsoby spustenia tejto procedúry (podľa spôsobu predávania parametrov) sú:

```
begin
add_dept;                                        -- použitie defaultných hodnôt
add_dept( 'TRAINING', 'NEW YORK');               -- pozíciou
add_dept( v_loc => 'DALLAS', v_name =>'EDUCATION'); -- názvom
add_dept( v_loc => 'BOSTON') ;                   -- názvom a default
end;
```

8.6.3 Zrušenie procedúry a funkcie

Syntax:

```
DROP PROCEDURE procedure_name;
DROP FUNCTION function_name;
```

Príklad:

```
SQL> DROP PROCEDURE raise_salary;
Procedure dropped.

SQL> DROP FUNCTION get_sal;
Function dropped.
```

8.6.4 Rozdiel medzi nepomenovaným a pomenovaným blokom

Hlavný rozdiel medzi nepomenovaným a pomenovaným blokom (procedúra, funkcia) je, že nepomenovaný blok sa vykonáva hneď po kompilácii zdrojového kódu a neukladá sa v databáze. Nepomenovaný blok sa pri každom spustení musí kompilovať a pri jeho spustení musíte mať priamo k dispozícii zdrojový kód daného bloku.

Pomenovaný blok príkazov (metóda) sa najprv skompiluje a jeho zdrojový aj skompilovaný kód sa uloží do databázového priestoru, pričom DBS vykoná operácie pre zefektívnenie práce servera. Teda metóda môže byť zavolaná z ľubovoľnej časti aplikácie a aj keď vymažeme súbor so zdrojovým kódom, pokiaľ nezmažeme samotnú metódu z DBS, bude ju možné naďalej používať.

8.7 Select v PL/SQL

Priamo v PL/SQL bloku je možné využiť len SELECT-INTO, alebo použiť kurzor pre spracovanie výsledkov selectu.

```
SELECT zoznam_stlpcov INTO zoznam_premennych
  FROM zoznam_tabuliek
  ...
```

Podmienky pre aplikovanie konštruktu SELECT-INTO

1. SELECT-INTO musí byť select, ktorý vráti práve jeden riadok.
2. Za INTO musí byť uvedený zoznam premenných, ktoré sú predtým deklarované.
3. Premenné musia v rovnakom poradí ako stĺpce selectu a rovnakého typu ako očakávame výsledok. Na to môžeme využiť kopírovanie typov(pomocou %TYPE).

Príklad 8.9: Select into

```
declare
  p_meno   os_udaje.meno%TYPE;
  p_priezv os_udaje.priezvisko%TYPE;
```

```
  pocet    integer;
begin
  select meno, priezvisko, count(*)
    INTO p_meno, p_priezv, pocet
  from os_udaje ou join student st using ( rod_cislo )
       left join zap_predmety using (os_cislo )
  where os_cislo = 550807
  group by meno, priezvisko;

  dbms_output.put_line('Pocet predmetov studenta - ' || p_meno || ' '||
                       p_priezv||' je '|| pocet );

end;
/
```

8.8 Kurzory

V tejto podkapitole uvedieme len jeden najjednoduchší spôsob spracovania výsledkov selectu v procedúre pomocou kurzoru.

1. Deklarácia kurzoru.

```
CURSOR nazov_kurzora [(nazov_parametra typ_parametra [,...])] IS
  select-prikaz;
```

2. Spracovanie riadkov vrátených kurzorom:

```
FOR nazov_recordu IN nazov_kurzoru[(hodnoty_parametrov)] LOOP

    .... príkazy s použitím premennej

       nazov_recordu.nazov_stlpca

END LOOP;
```

Príklad 8.10: Spracovanie pomocou kurzoru bez parametra

```
declare
  cursor cur1 IS
    select os_cislo, ou.meno, ou.priezvisko, st.st_skupina, sum( ects ) as kredity
      from os_udaje ou JOIN student st USING ( rod_cislo )
           JOIN zap_predmety zp USING ( os_cislo )
      group by os_cislo, ou.meno, ou.priezvisko, st.rocnik, st.st_skupina
      order by st.rocnik, st.st_skupina;

  i integer := 0;
  skupina char(6) := 'xxxxxx';
begin
  FOR st_rec IN cur1 LOOP
    if (skupina <> st_rec.st_skupina )then
      skupina := st_rec.st_skupina ;
      i:= 0;
      dbms_output.put_line(' --- Skupina: '|| skupina ||'   --- ');
    end if;
```

```
    i := i+1;

    dbms_output.put (i || ' '||st_rec.os_cislo || ', '||st_rec.meno  );
    dbms_output.put_line (' '|| st_rec.priezvisko || ' - ' || st_rec.kredity );
  END LOOP;
end;
```

Príklad 8.11: Spracovanie pomocou kurzoru s parametrami

```
declare
  cursor cur1 ( p_priezv VARCHAR, p_rocnik INTEGER ) IS
   select os_cislo, ou.meno, ou.priezvisko, st.st_skupina, sum( ects ) as kredity
      from os_udaje ou JOIN student st USING ( rod_cislo )
           JOIN zap_predmety zp USING ( os_cislo )
      where priezvisko like p_priezv
        and rocnik = p_rocnik
      group by os_cislo, ou.meno, ou.priezvisko, st.rocnik, st.st_skupina
      order by st.rocnik, st.st_skupina;

  i integer := 0;
  skupina char(6) := 'xxxxxx';
begin
  FOR st_rec IN cur1('K%', 3) LOOP
    if (skupina <> st_rec.st_skupina )then
      skupina := st_rec.st_skupina ;
      i:= 0;
      dbms_output.put_line(' --- Skupina: '|| skupina ||'   --- ');
    end if;

    i := i+1;

    dbms_output.put (i || ' '||st_rec.os_cislo || ', '||st_rec.meno  );
    dbms_output.put_line (' '|| st_rec.priezvisko || ' - ' || st_rec.kredity);
  END LOOP;
end;
```

Poznámka Metóda `dbms_output.put_line` vypisuje na obrazovku v prípade, že predtým použijeme zapnutie vypisovania:
SQL¿ set serveroutput on

!

8.9 Výnimky

- Štandardné výnimky

```
BEGIN
   prikazy bloku
EXCEPTION
  WHEN nazov_vynimky1 THEN
    priakzy1
  WHEN nazov_vynimky2 THEN
    priakzy2
END; -- ukoncenie bloku prikazov
```

- Užívateľom definované výnimky:

1. Pomocou funkcie

   ```
   RAISE_APPLICATION_ERROR(cislo_chyby, text_chyby);
   ```

   Číslo chyby musí byť z intervalu ¡-20000, -29999¿. Presne túto chybu nie je možné odchytiť v časti EXCEPTION, len medzi ostanými chybami (WHEN OTHERS THEN ...)

2. Pomocou premennej
   (a) Deklarácia chybovej premennej:
   ```
   chybova_premenna  EXCEPTION;
   ```
   (b) Vyvolanie výnimky:
   ```
   RAISE chybova_premenna;
   ```
   (c) Odchytenie výnimky:
   ```
   WHEN chybova_premenna THEN
     prikazy...
   ```

Príklad 8.12: Spracovanie štandardnej výnimky.

```
declare
  p_meno       os_udaje.meno%TYPE;
  p_priezv     os_udaje.priezvisko%TYPE;
BEGIN
  SELECT meno, priezvisko
       INTO p_meno, p_priezv
     from os_udaje
     where rod_cislo = '3';

  dbms_output.put_line( p_meno || ' '|| p_priezv);

EXCEPTION
  WHEN no_data_found THEN
    dbms_output.put_line ('Nebol najdeny');
  WHEN others THEN
    dbms_output.put_line ('Ina chyba');
END;
/
```

Exception	Oracle Error	SQLCODE Value
ACCESS_INTO_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

Príklad 8.13: Vyvolanie užívateľskej výnimky.

```
create or replace procedure
        zapis_predmet( oc  student.os_cislo%TYPE,
                         cp  predmet.cis_predm%TYPE,
                         rok zap_predmety.skrok%TYPE
                       )
AS
  pocet   integer;
BEGIN
  SELECT count(*) INTO pocet
    from student
    where os_cislo = oc;

  IF (pocet = 0) THEN
    RAISE_APPLICATION_ERROR(-2000, 'Not existing student');
  END IF;

  SELECT count(*) INTO pocet
    from predmet
    where cis_predm = cp;

  IF (pocet = 0) THEN
    RAISE_APPLICATION_ERROR(-2001, 'Not existing subject');
  END IF;


  SELECT count(*) INTO pocet
    from predmet_bod
    where cis_predm = cp
      and skrok = rok;

  IF (pocet = 0) THEN
    RAISE_APPLICATION_ERROR(-2002, 'Not existing subject for the year');
  END IF;

  INSERT INTO zap_predmety (os_cislo, cis_predm, skrok, prednasajuci, ects )
    SELECT oc, cp, rok, garant, ects
      FROM predmet_bod
      WHERE cis_predm = cp
        AND skrok = rok;

END;
/
```

Príklad 8.14: Vyvolanie a spracovanie užívateľskej výnimky.

```
create or replace procedure
        zapis_predmet( oc  student.os_cislo%TYPE,
                         cp  predmet.cis_predm%TYPE,
                         rok zap_predmety.skrok%TYPE
                       )
AS
  err1    EXCEPTION;
  err2    EXCEPTION;
  err3    EXCEPTION;
```

```
  pocet   integer;
BEGIN
  SELECT count(*) INTO pocet
    from student
    where os_cislo = oc;

  IF ( pocet = 0 ) THEN
    RAISE err1;
  END IF;

  SELECT count(*) INTO pocet
    from predmet
    where cis_predm = cp;

  IF ( pocet = 0 ) THEN
    RAISE err2;
  END IF;

  SELECT count(*) INTO pocet
    from predmet_bod
    where cis_predm = cp
      and skrok = rok;

  IF ( pocet = 0 ) THEN
    RAISE err3;
  END IF;

  INSERT INTO zap_predmety (os_cislo, cis_predm, skrok, prednasajuci, ects )
    SELECT oc, cp, rok, garant, ects
      FROM predmet_bod
      WHERE cis_predm = cp
        AND skrok = rok;

EXCEPTION
  WHEN err1 THEN
    dbms_output.put_line('Not existing student');
  WHEN err2 THEN
    dbms_output.put_line('Not existing subject');
  WHEN err3 THEN
    dbms_output.put_line('Not existing subject for the year');
  WHEN others THEN
    dbms_output.put_line('Other error');
END;
/

SQL> execute zapis_predmet ( 501555, 'BI06', 2015 );
Not existing subject for the year
```

8.9.1 Obmedzenia pre funkcie volané z príkazov DML

1. Funkcie volané v príkaze Select nemôžu obsahovať DML príkazy (Insert, Delete, Update).
2. Funkcie volané z príkazu Update alebo Delete nemôžu obsahovať ani príkazy Select, ani ostatné DML príkazy nad tou istou tabuľkou.
3. Funkcie volané z ľubovoľného SQL príkazu nemôžu obsahovať príkaz ukončenia transakcie (t.j., Commit, Rollback)

4. Funkcie volané z ľubovoľného SQL príkazu nemôžu obsahovať ani žiaden DDL príkaz (napr. Create Table), pretože DDL príkaz ukončí prebiehajúcu transakciu (generuje príkaz Commit).

5. Volanie podprogramov z funkcie, ktoré porušujú predchádzajúce obmedzenia je taktiež nepovolené.

Poznámka V prípade porušenia niektorého z uvedených obmedzení bude vyvolaná nasledovaná výnimka:

!

```
ORA-04091: .... is mutating, trigger/function may not see it
```

8.10 Zvýšenie kontroly prístupových práv (Invoker's rights)

V niektorých prípadoch je potrebné zvýšiť kontrolu prístupových práv na používané objekty. Jedným zo spôsobov je kontrola práv užívateľa, ktorý spúšťa príslušnú metódu a druhý spôsob kontroluje práva súvisiace s objektami, nad ktorými sa vykonávajú jednotlivé operácie. V ďalšom texte to podrobnejšie popíšeme a budeme demonštrovať na príkladoch.

8.10.1 Zvýšenie kontroly prístupových práv na objekty

Ak chceme kontrolovať práva používateľa, ktorý spúšťa príslušnú metódu, je nutné v tele metódy uviesť klauzulu `AUTHID CURRENT_USER`. Použitie tejto klauzuly spôsobí, že bude kontrolované, či používateľ, ktorý spúšta metódu, má práva na manipuláciu s objektami použitými v metóde.
Bez klauzuly `AUTHID CURRENT_USER` by táto kontrola nebola vykonaná, ale budú kontrolovať len práva na spustenie danej metódy (grant execute) a nie práva na samotné objekty.

Príklad 8.15: Metódy a práva

Na tomto príklade chceme ukázať, aké práva musí mať užívateľ, aby mohol úspešne spustiť metódu.

1. Užívateľ `kmat` vytvorí funkciu `vypis_hodnotenie` s klauzulou `AUTHID CURRENT_USER`, ktorá má vrátiť slovné hodnotenie podľa zadaného počtu bodov.

```
--- kmat
create or replace function vypis_hodnotenie ( pocet_bodov integer)
  return varchar2
     AUTHID CURRENT_USER
  AS
    vystup varchar2(35):='';
begin

   select vysledok|| ' - '|| slovne_hodnotenie
            INTO vystup
     from kmat.hodnotenie
   where pocet_bodov between hodnota_od and hodnota_do;

   return vystup;
end;
/
```

...t.j. ten, kto bude spúšťať túto procedúru, musí mať právo na tabuľku `kmat.hodnotenie`.

2. Užívateľ `kmat` pridelí právo na spustenie tejto funkcie užívateľovi `vajsova`.

```
grant execute on vypis_hodnotenie to vajsova;
```

3. Užívateľ `kmat` vypíše hodnotenie zodpovedajúce 92 bodom.

```
select kmat.vypis_hodnotenie(92) from dual;

VYPIS_HODNOTENIE(92)
--------------------------------------------------------------------------------
A - vyborne
```

4. Užívateľ `vajsova` sa teda pokúsi spustiť funkciu `kmat.vypis_hodnotenie(92)`.

```
--- vajsova
select kmat.vypis_hodnotenie(92) from dual
       *
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "KMAT.VYPIS_HODNOTENIE", line 8
```

...napriek tomu, že má právo na spustenie metódy, dostane chybové hlásenie, pretože nemá práva na select z tabuľky `kmat.hodnotenie`. (Všimnite si, že chybová hláška nezodpovedá reálnemu problému.)

5. Teda užívateľ `kmat` dá ešte práva na príkaz Select z tabuľky `kmat.hodnotenie` užívateľovi `vajsova`.

```
--- kmat
grant select on kmat.hodnotenie to vajsova;
```

6. Teraz sa užívateľ `vajsova` opäť môže pokúsiť o spustenie funkcie.

```
--- vajsova
SQL> select kmat.vypis_hodnotenie(92) from dual;

KMAT.VYPIS_HODNOTENIE(92)
--------------------------------------------------------------------------------
A - vyborne
```

8.10.2 Práva používateľa a schéma objektov

Štandardne pokiaľ v metóde neuvedieme schému objektov, bude nahradená schémou toho užívateľa, ktorý danú metódu vytvára. Ak však použijeme klauzulu `AUTHID CURRENT_USER` budeme musieť pamätať, že bude defaultne používaná schéma užívateľa, ktorý volá danú metódu. Výhodou je, že môžeme mať jednu metódu a každý užívateľ pracuje so svojimi vlastnými dátami.

Príklad 8.16: Práva používateľa (invoker) a defaultná schéma

1. Užívateľ `kmat` vytvorí funkciu `vypis_hodnotenie` s klauzulou `AUTHID CURRENT_USER`. Všimnite si, že tabuľku `hodnotenie` použil v príkaze Select bez schémy.

```
--- kmat
create or replace function vypis_hodnotenie ( pocet_bodov integer)
  return varchar2
    AUTHID CURRENT_USER
  AS
    vystup varchar2(35):='';
begin

   select vysledok|| ' - '|| slovne_hodnotenie
            INTO vystup
     from hodnotenie
   where pocet_bodov between hodnota_od and hodnota_do;
```

```
   return vystup;
end;
/
```

2. Dá práva na spustenie tejto funkcie užívateľovi `vajsova`.

```
--- kmat
grant execute on vypis_hodnotenie to vajsova;
```

3. Užívateľ `vajsova` sa pokúsi spustiť danú funkciu.

```
--- vajsova
SQL> select kmat.vypis_hodnotenie(92) from dual;
select kmat.vypis_hodnotenie(92) from dual
       *
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "KMAT.VYPIS_HODNOTENIE", line 8
```

4. Užívateľ `kmat` teda pridá práva na príkaz Select z tabuľky `hodnotenie`.

```
--- kmat
grant select on kmat.hodnotenie to vajsova;
```

5. Znovu sa užívateľ `vajsova` sa pokúsi spustiť danú funkciu, avšak bez úspechu.

```
--- vajsova
SQL> select kmat.vypis_hodnotenie(92) from dual;
select kmat.vypis_hodnotenie(92) from dual
       *
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "KMAT.VYPIS_HODNOTENIE", line 8
```

6. Užívateľ `vajsova` si musí vytvoriť tabuľku `hodnotenie` a naplniť ju dátami.

```
create table hodnotenie
(
   hodnota_od integer   check ( hodnota_od >= 0) NOT NULL,
   hodnota_do integer   check ( hodnota_do >= 0) NOT NULL,
   vysledok   char(1)   check ( vysledok in ('A','B','C','D','E','F'))
         NOT NULL      PRIMARY KEY,
   slovne_hodnotenie varchar2(30)
)
/

insert into hodnotenie values (95, 100, 'A', 'vyborne');
insert into hodnotenie values (90, 94, 'B', 'velmi dobre');
```

7. Až teraz bude funkcia správne pracovať.

```
--- vajsova
SQL> select kmat.vypis_hodnotenie(92) from dual;

KMAT.VYPIS_HODNOTENIE(92)
--------------------------------------------------------------------------------
B - velmi dobre
```

8. Všimnite si, že užívateľ `kmat` dostáva iné výsledky, aj keď používa tú istú funkciu s rovnakým vstupným parametrom. Je to dôsledok toho, že príkaz Select spracováva dáta z rôznych tabuliek.

```
--- kmat
SQL> select kmat.vypis_hodnotenie(92) from dual;

KMAT.VYPIS_HODNOTENIE(92)
--------------------------------------------------------------------------------
A - vyborne
```