Typy bloků PL/SQL

Materiály

- Školení PL/SQL se koná na Oracle DB Developer VM:
  - http://www.oracle.com/technetwork/community/developer-vm/index.html
  - https://www.virtualbox.org/

Základní struktura procedurálního kódu v PL/SQL – anonymní blok

- Kód jde psát přímo jako nepojmenovanou proceduru (anonymní blok):

```
[DECLARE]
  nepovinná deklarační sekce pro proměnné,
  konstanty, kurzory, datové typy
BEGIN
  povinná výkonná sekce, jde sem případně
  zanořit i další anonymní blok...
[EXCEPTION
  nepovinná sekce pro zpracování výjimek]
END;
```

Procedura vs. funkce

- Procedury a funkce jsou pojmenované bloky PL/SQL kódu:
  - Funkce:
    - Má povinnou návratovou hodnotu.
    - Může se použít jako skalární funkce, nebo se volá v PL/SQL bloku.
    - Může volat SELECTy, NEMŮŽE volat DML příkazy (INSERT, UPDATE, DELETE).
  - Procedura:
    - Nemá žádnou návratovou hodnotu
    - Volá se pouze pomocí PL/SQL bloku.
    - Může volat SELECTy i DML příkazy

Procedura – syntaxe

```
CREATE [OR REPLACE] PROCEDURE <název procedury>
[(<seznam parametrů>)] AS
  deklarační sekce
BEGIN
  výkonná sekce
[EXCEPTION
  sekce pro zpracování výjimek]
END;
```

```
<seznam parametrů> = <jméno parametru1>
  [{IN|OUT|IN OUT}] <typ> [DEFAULT <hodnota>], ...
```

- Pokud uvedeme OR REPLACE, již existující procedura stejného jména se může přepsat (jinak by se proceduru se stejným názvem nepodařilo vytvořit).
- Smazání procedury: DROP PROCEDURE <název procedury>;

Procedura – příklad

```
CREATE OR REPLACE PROCEDURE zvyseni_mzdy (procento NUMBER) AS
BEGIN
  UPDATE zamestnanec SET mzda = mzda * (1 + procento/100);
END;
```

spuštění procedury:

```
EXECUTE zvyseni_mzdy(6);
```

nebo:

```
BEGIN
  zvyseni_mzdy(6);
END;
```

Poznámka: Parametry musíme předávat buď v takovém pořadí, v jakém jsou definovány, nebo v libovolném pořadí při použití této syntaxe:

```
ZVYSENI_MZDY(PROCENTO => 10);
```

Funkce – syntaxe

```
CREATE [OR REPLACE] FUNCTION <název funkce> [(<seznam parametrů>)] RETURN <datový typ výsledku> AS
  deklarační sekce
BEGIN
  výkonná sekce
  RETURN <hodnota>; -- vrácení hodnoty z funkce
[EXCEPTION
  sekce pro zpracování výjimek]
END;
```

```
<seznam parametrů> = <jméno parametru1>
  [{IN|OUT|IN OUT}] <typ> [DEFAULT <hodnota>], ...
```

- Smazání funkce: DROP FUNCTION <název funkce>;

Funkce - příklad

```
create or replace function format_jmeno(jmeno varchar2, prijmeni varchar2)
                                    return varchar2 as
begin
  return initcap(trim(jmeno)) || ' ' || upper(trim(prijmeni));
end;
```

Spuštění funkce:

Buď jako skalární funkce:

```
select format_jmeno(jmeno, prijmeni) from zamestnanec;
```

Nebo v PL/SQL kódu:

```
declare
  jmeno varchar(255) := 'jirka';
  prijmeni varchar(255) := 'pinkas';
  formatted_jmeno varchar(255);
begin
  formatted_jmeno := format_jmeno(jmeno, prijmeni);
  dbms_output.put_line('vysledek: ' || formatted_jmeno);
end;
```

IN, OUT, IN OUT

- V předcházejícím příkladu jsem argumenty funkce definoval následovně:

```
create or replace function format_jmeno(jmeno varchar2, prijmeni varchar2)
```

- Mohl jsem také napsat:
- `create or replace function format_jmeno(jmeno IN varchar2, prijmeni IN varchar2)`
- Pokud neuvedeme jakého je argument typu, pak se jedná o argument typu IN. Existují i další:
  - IN: Výchozí, read-only. Uvnitř procedury / funkce není možné změnit hodnotu takového parametru (vyhodí se chyba kompilace).
  - OUT: write-only. Není možné nastavit hodnotu OUT parametru zvenčí.
  - IN OUT: read-write.

Trigger

- Trigger – procedurální kód, který lze spouštět automaticky před/namísto/po každém provedení příkazu INSERT, DELETE nebo UPDATE na vybrané tabulce. Trigger je navázán na tabulku, při zrušení tabulky příkazem DROP je zrušen i trigger.

```
CREATE [OR REPLACE] TRIGGER <název triggeru> {BEFORE|
AFTER|INSTEAD OF} {INSERT|DELETE|UPDATE} ON <název
tabulky> [FOR EACH ROW [WHEN <podmínka>]]
[DECLARE
  deklarační sekce]
BEGIN
  výkonná sekce
[EXCEPTION
  sekce pro zpracování výjimek]
END;
```

DBMS_OUTPUT.PUT_LINE

- Jak zobrazit testovací výpisy? V SQL Developer otevřete View → Dbms Output a klikněte na velké zelené PLUS (Enable DBMS OUTPUT for Connection) a otevřete příslušné databázové připojení.
- Poté již můžete normálně spouštět Vaše kódy pomocí Run statement (Ctrl + Enter).