Úvod do PL/SQL 1

Ing. Pavol Sojka, Katedra aplikovanej informatiky, FHI, EU

PL/SQL je procedurálne rozšírenie jazyka SQL vyvinuté spoločnosťou Oracle. Je to jazyk transakčne orientovaný, platformovo nezávislý a teda prenosný. Okrem databázy Oracle bol implementovaný aj do databázy DB2 od firmy IBM.

PL/SQL dáva programátorom možnosť robiť zložité databázové dotazy, robiť náročné transformácie dát. Používa veľa dátových typov, prístup k preddefinovaným SQL balíkom (packages) a iné.

PL/SQL nie je samostatný programovací jazyk, ale je súčasťou komplexného vývojového prostredia Oracle. Štandardným nástrojom na prístup do PL/SQL je textové rozhranie SQL*Plus. SQL*Plus umožňuje zadávať interaktívne príkazy aj spúšťanie skriptov v jazyku SQL aj PL/SQL. Pre veľké bloky kódu v PL/SQL je vhodnejšie použitie externých editorov, ako napr. Notepad++, PSPad a pod. Skripty sa v SQL*Plus spúšťajú v podobe @meno_skriptu.sql, teda zavináč indikuje rozhraniu SQL*Plus, ze sa ním myslí súbor s príkazmi.

Základná syntax jazyku PL/SQL je nasledovná:

```
DECLARE
  <deklarácia premenných>
BEGIN
  <príkazy>
EXCEPTION
  <ošetrenie chýb, tzv. výnimiek>
END;
```

jednotlivé príkazy jazyka PL/SQL sa oddeľujú bodkočiarkou.

1. Jednoduchý príklad na výpis „Hello world“

```
DECLARE
  msg varchar2(20):= 'Hello World';
BEGIN
  dbms_output.put_line(msg);
END;
/
```

Tento jednoduchý skript vypíše obsah premennej „msg“, do ktorej sa predtým uloží textový reťazec „Hello World“. SQL*Plus môže v určitých situáciach vyžadovať vloženie lomítka na konci programového zápisu, čo signalizuje DB serveru, že písanie kódu je ukončené a skript sa môže vykonať.

2. Základné dátové typy

Číselné (integer, float, double,...), reťazce (char, varchar, long,...), logické (boolean – true, false, null)*****, časové (year, day, month,...).

*****boolean premenné pozná a spracúva len PL/SQL, čisté SQL ich definície nepozná a preto sa v SQL nepoužívajú a ani vo vnútorných volaniach SQL príkazov z PL/SQL. NULL je špecifická forma hodnoty, ktorá

reprezentuje chýbajúce alebo žiadne dáta. Hodnota NULL môže byť priradená alebo vložená do DB, ale nemôže byť súčasťou žiadnych výpočtov.

Špeciálne formáty v PL/SQL:

- BLOB – binary large object – umožňuje ukladanie binárnych dát **v databáze** až do veľkosti 128 TB (terabajtov). BLOB môže ukladať texty, obrázky, video, hudbu priamo v DB.

- BFILE – ukladá binárne objekty mimo DB, teda sú uložené v súborovom systému príslušného operačného systému. Ich maximálna veľkosť je závislá od verzie a typu operačného systému, ale maximálna veľkosť tohto typu súboru by nemala prekročiť 4 GB.

- CLOB – character large object - umožňuje ukladanie veľkých blokov textových dát v databáze až do veľkosti 128 TB.

- NCLOB - National Character Large Object - umožňuje ukladanie veľkých blokov textových dát v databáze až do veľkosti 128 TB v rôznych jazykových mutáciách znakových sád (čínske, arabské znaky, azbuka a pod.)

Úvod do PL/SQL 2

Ing. Pavol Sojka, Katedra aplikovanej informatiky, FHI, EU

Premenné

Premenné v PL/SQL sú definované ako oblasti v pamäti počítača, kde PL/SQL uchováva hodnoty dátových typov, napr. čísla, reťazce a pod. Premenné môžu byť definované malými aj veľkými písmenami a sú teda tzv. case-insensitive, teda premenná ***meno*** môže mať aj tvar ***MENO*** aj ***Meno***. Maximálna dĺžka názvu premenných by nemala presiahnuť 30 znakov a nemala by sa volať ako príkaz PL/SQL (tzv. rezervované kľúčové slovo), teda premenná nemôže byť napr. BEGIN alebo END a pod.

Príklad definície premenných:

cena number(10, 2);
pi CONSTANT double precision := 3.1415;
meno varchar2(25);
adresa varchar2(100);

Premenné sú automaticky inicializované ako NULL, pokiaľ im nie sú priradené v deklaračnej časti skriptu východzie premenné definované ako ***DEFAULT*** alebo priradené pomocou kombinácie znakov ***:=***

pocitadlo binary_integer := 0;
odkaz varchar2(30) DEFAULT 'Skusobny text';

Ak je premenná inicializovaná s parametrom NOT NULL, teda nemôže mať východziu hodnotu NULL, tak musí mať explicitne (programátorom) definovanú hodnotu, viď napr. premenná ***pocitadlo*** má východziu hodnotu nula.

Je vhodnou praxou definovať také množstvo východzích premenných ako je možné.

Príklad:

```
DECLARE

  a integer := 100;
  b integer := 200;
  c integer;
  d real;

BEGIN
  c := a + b; --100+200
  dbms_output.put_line('Hodnota c: ' || c);
  /* Desatinne
          cisla */
  d := 80.0/7.0;
  dbms_output.put_line('Hodnota d: ' || d);
END;
/
```

Horeuvedený príklad spočíta hodnoty a+b a uloží ich do premennej c, ktorú vypíše do konzoly (napr. SQL*Plus). Symbol || znamená zreťazenie, teda pridaj výsledok výpočtu za reťazec ***'Hodnota c: '*** alebo ***'Hodnota d: '.***

Dve pomlčky, resp. dva mínusy za sebou znamenajú jednoriadkový komentár (single-line comment), ktorý znamená, že za mínusmi sa vložené slová neberú ako príkazy, ale ako sprievodný komentár k príkazu a môže v ňom byť čokoľvek a berie sa ako komentár po vložení nového riadku – enteru.

Komentár typu /* ..hocičo.. */ je komentár, ktorý sa otvára znakmi **/*** a môže sa do nich vložiť rôzne dlhý text/komentár oddelený hocakým počtom nových riadkov – enterov. Komentár sa ukončuje znakmi */. Nič medzi týmito otváracími a zatváracími značkami sa neberie ako programový kód.

4. Kontext platnosti premenných v PLSQL

PLSQL umožňuje tzv. vnáranie jednotlivých blokov do seba. Nadradený blok môže čítať aj prepisovať premenné vo vnorenom (vnútornom, nested) bloku, ale naopak vnútorný blok nemôže, resp. „nevidí“ premenné nadradeného bloku. Toto sa nazýva v anglickej terminológii *Variable scope.* Teda scope sa chápe ako oblasť, priestor, dosah a pod. kde môže byť s premennými manipulované. Na základe tohto sa delia premenné na globálne, ktoré sú definované v najvyššom bloku sú viditeľné pre všetky vnútorné (vložené) bloky a lokálne, ktoré sú definované vo vnútorných blokoch a nie sú viditeľné pre vonkajšie (nadradené) bloky.

Príklad:

```
DECLARE
  -- Globálne premenné
  num1 number := 1;
  num2 number := 2;
BEGIN
  dbms_output.put_line('Vonkajsia premenna num1: ' || num1);
  dbms_output.put_line('Vonkajsia premenna num2: ' || num2);
  DECLARE
    -- Lokálne premenné
    num1 number := 3;
    num2 number := 4;
  BEGIN
    dbms_output.put_line('Vnutorna premenna num1: ' || num1);
    dbms_output.put_line('Vnutorna premenna num2: ' || num2);
  END;
END;
/
```

Okrem priameho priradenia premennej (napr. num:=1) môžeme premenné priraďovať aj priamo cez SQL dotaz z tabuľky, vid napr. tabulka zamestnancov:

do premenných **meno** a **pozicia** uložíme priamo údaje z tabuľky v DB zamestnanca s číslom 7566

```
DECLARE
 meno varchar(50);
 pozicia varchar(50);
BEGIN
 select ename, job into meno, pozicia from emp where empno=7566;
```

```
dbms_output.put_line ('Meno a pozicia: ' || meno || ', ' || pozicia);
END;
/
```

Premenné a operátory v PL/SQL

Ing. Pavol Sojka, Katedra aplikovanej informatiky, FHI, EU

1. Deklarovanie konštánt a premenných

Konštanta je v jazyku PL/SQL definovaná kľúčovým slovom **CONSTANT.** Premenná tohto typu si vyžaduje explicitnú definíciu, teda musí mať pridelenú hodnotu už pri inicializácií a nie je možné ju už za behu programu meniť.

Príklad pri definícii konštanty:

PI CONSTANT NUMBER := 3.1415

Pri deklarovaní môžu premenné nadobúdať hodnoty číslo, znak, reťazec, logické hodnoty true/false, dátumové hodnoty.

2. Operátory v PL/SQL (pre príklady je použitá schéma používateľa SCOTT**)**

Základné operátory sú operátory relačné operátory, operátory porovnania, priradenia a logické operátory.

Relačné operátory:

A=B; A<>B, A!=B; >; <; >=; <=

Operátory porovnania:

LIKE - porovnáva cieľový reťazec s hľadaným vzorom. Používa sa aj so znakom %, ktorý zabezpečí, že sa hľadaný reťazec vyhľadáva ako časť textu.

select * from dept where dname like '**SALES**'; - vráti presne zadaný reťazec
select * from dept where dname like '**SALE%**'; - vráti čokoľvek, čo začína SALE
select * from dept where dname like '**%ALE%**'; - vráti čokoľvek, čo obsahuje ALE
select * from dept where dname like '**%ALES**'; - vráti čokoľvek, čo končí na ALES

BETWEEN – vráti hodnoty, ktoré sú v rozsahu aj vrátane hraničných hodnôt

select * from SALGRADE where losal between 700 and 1401; - vráti mzdy v rozpätí 700-1401 vrátane hornej aj spodnej hranice

IN – vráti hodnoty, ktoré sa nachádzajú v udanom rozsahu

select * from dept where LOC IN ('DALLAS', 'PRAHA', 'BOSTON', 'ROME'); - vráti z databázy riadky, ktoré v stĺpci LOC obsahujú uvedené mestá.

IS NULL – vráti z databázy riadky, ktoré v stĺpcoch, ktoré tetsujeme, majú hodnotu NULL

vloženie nového riadku do DB s hodnotou NULL:
insert into dept (DEPTNO,DNAME,LOC) values (50, 'SKLAD', NULL);

overenie, vrátenia hodnoty NULL:
select * from dept where LOC IS NULL;

Logické operátory:

AND – porovnáva dve a viac podmienok, ktoré musia platiť súčasne, napr.:

select ename, job from emp where mgr=7698; - vráti všetkých zamestnancov, ktorých manažér je osoba s číslom 7698

select ename, job from emp where mgr=7698 ***AND*** *job='SALESMAN';* - vráti všetkých zamestnancov, ktorých manažér je osoba s číslom 7698 a zároveň je zamestnanec predajca (SALESMAN).

OR - porovnáva dve a viac podmienok, ktoré nemusia platiť súčasne, napr.:

select ename, job, mgr from emp where mgr=7698 ***OR*** *job='CLERK';* - vráti zamestnancov, ktorých manažér je osoba s číslom 7698 alebo majú popis práce úradník (clerk).

NOT – neguje zadanú podmienku

select ename, job, mgr from emp where mgr=7698 and job='CLERK'; - vráti zamestnancov, ktorých manažér je osoba s číslom 7698 a zároveň **majú** popis práce úradník (clerk).

select ename, job, mgr from emp where mgr=7698 and ***NOT*** *job='CLERK';* - vráti zamestnancov, ktorých manažér je osoba s číslom 7698 a zároveň **nemajú** popis práce úradník (clerk).

Podmienky v PL/SQL

Ing. Pavol Sojka, Katedra aplikovanej informatiky, FHI, EU

Podmienky v PL/SQL

PL/SQL poskytuje možnosti pre vyhodnocovanie podmienok, a to IF – THEN a CASE štruktúry. Jednotlivé štruktúry môžu mať nasledujúce podoby:

a) IF-THEN

IF podmienka THEN

    kód, ak podmienka platí

END IF;

b) IF-THEN-ELSE

IF podmienka THEN

    kód sa vykoná, ak podmienka platí. Nasledujúce ELSE sa už nevykoná.

ELSE

    kód sa vykoná, ak predchádzajúca podmienka neplatí

END IF;

c) IF-THEN-ELSIF

Tento typ podmienky dovolí testovať viac premenných naraz v jednom príkaze IF.

IF podmienka THEN

    Podm1 - kód sa vykoná, ak podmienka platí. Nasledujúce testy už nevykoná.

ELSIF

    Podm2 - kód sa vykoná, ak podmienka platí. Nasledujúce testy už nevykoná.

ELSIF

    Podm3 - kód sa vykoná, ak podmienka platí. Nasledujúce testy už nevykoná.

ELSE

    kód sa vykoná, ak predchádzajúce podmienky neplatia

END IF;

Pozor, príkaz je **ELSIF** nie ELSEIF. ELSE nasleduje až po poslednom ELSIF-e.

d) Príkaz CASE

```
CASE premenna
  WHEN 'hodnota1' THEN prikaz1;
  WHEN 'hodnota2' THEN prikaz2;
  WHEN 'hodnota3' THEN prikaz3;
  ...
  ELSE prikazN;  -- default case
END CASE;
```

Štruktúra CASE … END CASE používa tzv. výberové kritérium (selector) – v našom prípade *premenná*, kde sa riadok po riadku vyhodnocuje, či sa premenná (selector) zhoduje s kritériami za kľúčovým slovom WHEN. Ak sa zhoduje, tak sa vykonajú príkazy za THEN. Ak sa nezhoduje ani jedna hodnota, tak sa použije hodnota za príkazom ELSE (tzv. default case). V CASE sa hľadá jedna vyhovujúca hodnota, ostatné sa už potom nevykonajú.

e) Príkaz CASE (prehľadávaný)

Tento CASE je podobný predchádzajúcemu, len s tým, že pri overovaní kritérií nepoužíva tzv. premenná selector za CASE, ale priamo za kľúčovým slovom WHEN, kde sa porovnávajú kritériá s prehľadávanými hodnotami. Výhodou je, že je možné použiť pre vyhľadávanie viac kritérií.

```
CASE
  WHEN premenna = 'hodnota1' THEN prikaz1;
  WHEN premenna = 'hodnota2' THEN prikaz2;
  WHEN premenna = 'hodnota3' THEN prikaz3;
  ...
  ELSE prikazN;  -- default case
END CASE;
```

f) Vložené (nested) IF-THEN-ELSE

```
IF( podmienka 1)THEN
  -- vykona sa, ked je podmienka 1 pravdiva (true)
  IF(podmienka 2) THEN
    -- vykona sa, ked je podmienka 2 pravdiva (true)
    prikazy pre podmienku 2;
  END IF;
```

```
ELSE
  -- vykona sa, ked je podmienka 1 nie je pravdiva (false)
  prikazy po ELSE pre podmienku 1, ak nie je pravdiva (false);
END IF;
```

Vložený IF sa vykonáva **vo vnútri** nadradeného IF-u.

Cykly v PL/SQL

Ing. Pavol Sojka, Katedra aplikovanej informatiky, FHI, EU

Cykly (LOOPS)

Cykly zabezpečujú vykonanie zvoleného bloku kódu viac krát. Keďže príkazy v programovom kóde sú vykonávané sekvenčne (jeden za druhým), je niekedy vhodné zmeniť, resp. oneskoriť poradie vykonávaných príkazov a niektoré spustiť podľa potreby viac krát a to pomocou cyklu. Po skončení cyklu sa spustia tie „oneskorené“ - mimo cyklu, lebo čakali až skončí cyklus príkazov pred nimi.

a) Základný LOOP v PL/SQL

Základná LOOP štruktúra obsahuje medzi príkazmi LOOP … END LOOP sekvenciu príkazov, ktoré sa majú v cykle vykonávať dovtedy, pokiaľ nie je splnená podmienka ukončenia cyklu.

```
LOOP
  Sekvencia príkazov;
END LOOP;
```

Keďže LOOP … END LOOP je nekonečný cyklus, musíme pre jeho skončenie určiť podmienku, kedy sa má cyklus ukončiť. Podmienku určujeme pomocou IF-THEN alebo pomocou kľúčových slov EXIT WHEN (viď príklady).

b) WHILE ... LOOP

Cyklus podobný predchádzajúcemu, ktorý má ale podmienku pre ukončenie cyklu definovanú hneď na začiatku.

```
WHILE podmienka LOOP
  Sekvencia príkazov;
END LOOP;
```

(viď príklad)

c) FOR ... LOOP

Cyklus s konečným počtom opakovaní. Hodnota pocitadla sa môže vždy zvýšiť len o 1.

```
FOR pocitadlo IN startovacia hodnota .. konecna hodnota LOOP
    Sekvencia príkazov;
END LOOP;
```

pozn.: Ak je potrebné použiť zvyšovanie hodnoty napr. o dve, dá sa to spraviť pomocou príkazu mod(pocitadlo,2) alebo pomocou cyklov vyššie (LOOP .. END LOOP, WHILE LOOP … END LOOP).

```
IF MOD(pocitadlo,2) = 0 THEN
  DBMS_OUTPUT.PUT_LINE('Skok o 2 ' || pocitadlo);
END IF;
```

d) Reverzný FOR ... LOOP

Reverzný cyklus FOR začína na konečnej hodnote a klesá k počiatočnej. Zápis je rovnaký ako v predchádzajúcom príklade len s tým rozdielom, že sa za kľúčové slovo IN dá kľúčové slovo REVERSE. Teda štartovacia ani konečná hodnota sa nemenia, napr. FOR i IN **REVERSE** 1 .. 10.

```
FOR pocitadlo IN REVERSE startovacia hodnota .. konecna hodnota LOOP
    Sekvencia príkazov;
END LOOP;
```

e) Vnorené (nested) cykly

Jednotlivé cykly môžu byť vnárané do seba. Vždy končí ako prvý vnútorný cyklus.

```
LOOP
  Sekvencia príkazov;
  LOOP
    Sekvencia príkazov;
  END LOOP;
END LOOP;
```

f) Pomenovávanie jednotlivých cyklov (labeling)

Jednotlivé cykly v PLSQL môžu byť pomenované. Je to praktické nielen pre lepšie rozlíšenie, čo daný cyklus vykonáva, ale môže toto pomenovanie byť použité ako súčasť iných príkazov, napr. pri príkaze na opustenie cyklu podľa mena (napr. ak je viac cyklov do seba vnorených).

Menovka (label) cyklu ma tvar **<< menovka >>** a dáva sa na začiatok LOOP cyklu.

```
<< cyklus1 >>
LOOP
  Sekvencia príkazov;
END LOOP cyklus1;
```

g) Príkazy pre kontrolu opustenia cyklu

EXIT – opustenie cyklu a pokračovanie vo vykonávaní ďalších príkazov za cyklom alebo opustenie vnútorného cyklu a pokračovanie vo vykonávaní nadradeného (vonkajšieho) cyklu.

EXIT WHEN – opustenie cyklu na základe podmienky (viď príklady vyššie).

CONTINUE – tento príkaz v cykle má za úlohu pri splnení zadanej podmienky ignorovať zbytok kódu za ním a hneď skočiť na začiatok cyklu a pokračovať v jeho vykonávaní (viď príklad).

GOTO – Príkaz GOTO umožňuje v rámci programu skočiť do ľubovoľnej časti kódu, ktorá je označená menovkou (návestie, label). Tento príkaz nie je odporúčané používať, lebo zbytočne komplikuje svojimi volaniami beh programu.

Použitie:

GOTO menovka;

..

<< menovka >>

príkazy;