Systemy baz danych - wykład III

dr inż. Robert Perliński

Instytut Informatyki Teoretycznej i Stosowanej
Politechnika Częstochowska

Październik 2015

Plan wykładu

- Uzupełnienie projektu fizycznego
- Czym jest PL/SQL
- Bloki w PL/SQL
- Zmienne i stałe
- Instrukcje sterujące
- Wykorzystanie SQL w PL/SQL
- Kursory
- Wyjątki

Projekt fizyczny bazy danych

Braki projektu fizycznego opartego wyłącznie na możliwościach SQL'a:

- niemożność modyfikacji danych za pośrednictwem perspektyw złożonych,
- brak obsługi sytuacji wyjątkowych,
- brak obsługi zdarzeń,
- brak funkcji i procedur użytkownika,
- brak złożonych zależności i ograniczeń integralności.

PL/SQL

(ang. Procedural Language/Structured Query Language) - język programowania będący proceduralnym rozszerzeniem SQL wprowadzonym przez firmę Oracle (oparty na języku Ada). Umożliwia wykorzystywanie konstrukcji takich jak pętle, instrukcje warunkowe, zmienne oraz tworzenie procedur, funkcji, wyzwalaczy użytkownika wspomagających konkretną realizację bazy danych.

Rozwiązanie specyficzne dla SZBD Oracle, jednakże każdy dystrybutor udostępnia swoje rozwiązania (np. PL/pgSQL w PostgreSQL), które są zbliżone do PL/SQL'a.

Cechy PL/SQL

Cechy PL/SQL:

- język proceduralny nastawiony na przetwarzanie danych,
- język kompilowany,
- język obsługujący zmienne, stałe, struktury sterujące, wyjątki,
- ma strukturę blokową,
- każda instrukcja wewnątrz bloku PL/SQL jest zakończona średnikiem,
- operator przypisania to `:=` ,
- może zawierać instrukcje DML i TCL,
- nie może zawierać instrukcji DDL i DCL (nadawanie uprawnień).

Bloki PL/SQL

Podstawową jednostką, z której składa się program w PL/SQL jest blok, wyróżniamy:

- bloki anonimowe,
- bloki nazwane,
- bloki podrzędne.

Każdy blok składa się z maksymalnie trzech części:

- deklaracyjnej (opcjonalnie),
- wykonywalnej,
- obsługi wyjatków (opcjonalnie).

Struktura bloku PL/SQL

```
[DECLARE
  deklaracje;]
BEGIN
  bloki_podrzedne;
  instrukcje;
  [EXCEPTION
    obsluga_wyjatkow;]
END;
```

Komentarze

W PL/SQL stosuje się dwa rodzaje komentarzy:

- jednowierszowe - rozpoczynające się od znaków --
- wielowierszowe - rozpoczynające się od znaków /*, a kończące */.

Typy danych

PL/SQL obsługuje wszystkie typy danych SQL, oraz posiada wiele swoich, dodatkowych typów, które możemy podzielić następująco:

- typy proste (BINARY_INTEGER, NATURAL, BOOLEAN, POSITIVE, itd.),
- typy złożone (RECORD, TABLE, VARRAY),
- typy referencyjne (REF CURSOR, REF typ_obiektowy),
- typy definiowane przez użytkownika.

Zmienne

Deklaracja zmiennych:

```
DECLARE
  nazwa_zmiennej typ[(rozmiar)]
   [NOT NULL] [{:= | DEFAULT} wartosc];
```

Np.:

```
DECLARE
  v_wiek NUMBER(3);
  v_plec VARCHAR2(9);
  v_licznik NUMBER(4) NOT NULL := 0;
  v_suma NUMBER(10) DEFAULT 0;
```

Zmienne rekordowe

Deklaracja zmiennych rekordowych:

```
DECLARE
  TYPE nazwa_typu_rekordowego IS RECORD (
    pole1 typ [inicjalizacja]
    [, pole2 typ [inicjalizacja] ...]
  );
  nazwa_zmiennej typ_rekordowy;
```

Np.:

```
DECLARE
  TYPE r_adres IS RECORD (
    v_ulica VARCHAR2(30),
    v_nr_domu VARCHAR2(4),
    v_nr_mieszkania VARCHAR2(4)
  );
  vr_adres r_adres;
```

Stałe

Deklaracja stałych:

```
DECLARE
  nazwa_stalej CONSTANT typ[(rozmiar)]
    {:= | DEFAULT} wartosc;
```

Np.:

```
DECLARE
  c_procent CONSTANT NUMBER(2) := 10;
```

%TYPE, %ROWTYPE

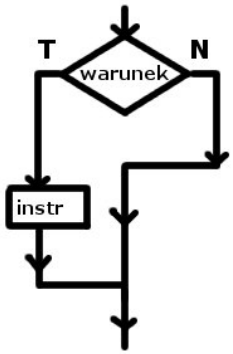
Istnieje możliwość deklarowania zmiennych o typach kolumn, lub wierszy istniejących w bazie:

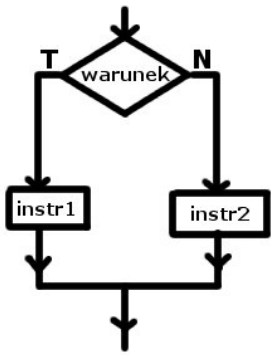
```
DECLARE
  nazwa_zmiennej tabela.kolumna%TYPE;
  nazwa_z_rekordowej tabela%ROWTYPE;
```

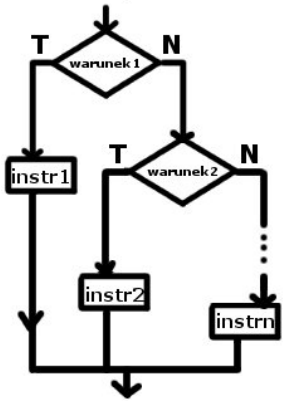
Np.:

```
DECLARE
  v_nazwisko studenci.nazwisko%TYPE;
  vr_student studenci%ROWTYPE;
```

IF





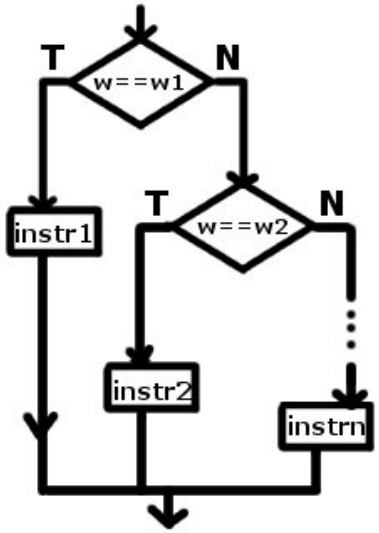


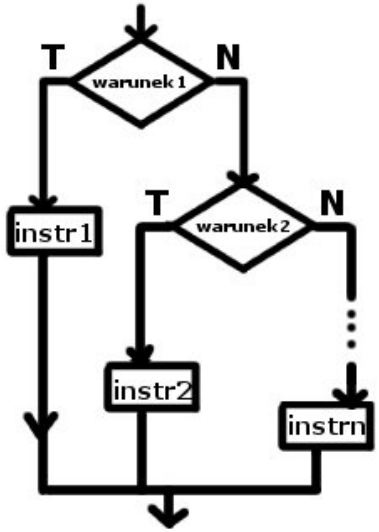

```
IF warunek THEN
  instrukcja;
END IF;

IF warunek THEN
  instrukcja1;
ELSE
  instrukcja2;
END IF;

IF warunek1 THEN
  instrukcja1;
ELSIF warunek2 THEN
  instrukcja2;
...
ELSE
  instrukcja_n;
END IF;
```

CASE

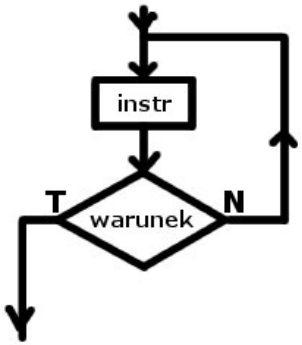


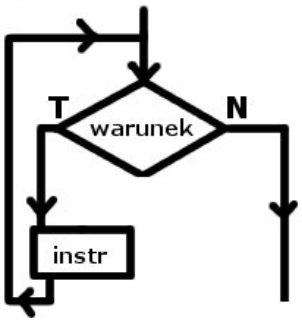


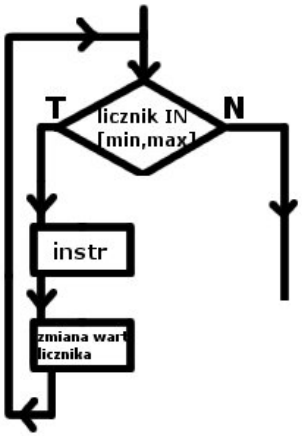

```
CASE wyrazenie
  WHEN wyrazenie1 THEN instrukcja1;
  WHEN wyrazenie2 THEN instrukcja2;
  ...
  [ELSE instrukcja_n];
END CASE;
```

```
CASE
  WHEN warunek1 THEN instrukcja1;
  WHEN warunek2 THEN instrukcja2;
  ...
  [ELSE instrukcja_n];
END CASE;
```

Pętle








```
LOOP
  instrukcje;
  [IF warunek THEN EXIT; END IF;]
  [EXIT WHEN warunek;]
END LOOP;

WHILE warunek LOOP
  instrukcje;
END LOOP;

FOR licznik IN [REVERSE] min..max LOOP
  instrukcje;
END LOOP;
```

Instrukcja NULL

```
BEGIN
  IF warunek THEN
    instrukcje;
  ELSE
    NULL;
  END IF;
END;
```

Interakcje

- pobieranie danych (zmienne podstawienia):

```
v_nazwisko := &nazwisko;
```

- wyprowadzenie danych (włączona zmienna środowiskowa SERVEROUTPUT ON):

```
dbms_output.put_line(tekst);
dbms_output.new_line;
dbms_output.put_line(
    'Hello'||CHR(10)||' World');
```

- wyczyszczenie ekranu (nie działa w PL/SQL):

```
CLEAR SCREEN;
CL SCR;   -- skrócona forma
```

SQL w PL/SQL I

Wewnątrz bloków PL/SQL można zagnieździć niektóre polecenia SQL:

- DML: INSERT, UPDATE, DELETE, specyficzną formę SELECT,
- TCL: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION, LOCK.

Postać klauzuli SELECT:

```
SELECT lista_wyrazen INTO lista_zmiennych
  FROM lista_relacji...;

SELECT lista_wyrazen INTO zmienna_rekordowa
  FROM lista_relacji...;
```

SQL w PL/SQL II

Postać INSERT, UPDATE i DELETE pozostaje niezmieniona, dochodzi opcjonalna klauzula `RETURNING wyrazenie INTO zmienna`:

```
INSERT INTO dzialy (id_dzialu, siedziba)
  VALUES (90, 'HONOLULU')
  RETURNING id_dzialu INTO v_id;
```

Istnieje możliwość wykorzystania zmiennych rekordowych:

```
INSERT INTO tabela VALUES zmienna_rekordowa;
UPDATE tabela SET row = zmienna_rekordowa;
```

Kursor I

Kursor

- przestrzeń robocza posiadająca nazwę, pozwalająca składować i udostępniać wynik zapytania SQL. Wyróżniamy kursory jawne (deklarowane przez programistę) i niejawne o nazwie SQL (tworzone przez system dla każdej instrukcji DML i DDL).

Obsługa kursora jawnego:

- deklaracja,
- otwarcie,
- pobranie danych,
- zamknięcie.

Deklaracja kursora I

Deklaracja kursora:

```
CURSOR nazwa [(parametr1 typ [:=|DEFAULT wartosc1]
              [, parametr2...]...)]
   [RETURN typ] IS SELECT...;
```

Przykłady:

```
CURSOR c_szefowie IS
  SELECT DISTINCT kierownik FROM pracownicy;

CURSOR c_dzial (p_id NUMBER) IS
  SELECT * FROM pracownicy WHERE id_dzialu=p_id;

CURSOR c_dzial (p_id NUMBER := 10) IS
  SELECT * FROM pracownicy WHERE id_dzialu=p_id;

CURSOR c_pracownicy RETURN pracownicy%ROWTYPE;
```

Otwarcie kursora

```
OPEN nazwa (lista_parametrow);
```

Np.:

```
DECLARE
  CURSOR c_dzial (p_id NUMBER := 10) IS
    SELECT * FROM pracownicy WHERE id_dzialu=p_id;
  ...
BEGIN
  OPEN c_dzial(20);
  ...
END;
```

Pobranie danych

```
FETCH nazwa INTO {lista_zmiennych|zmienna_rekordowa};
```

Np.:

```
DECLARE
  CURSOR c_dzial (p_id NUMBER := 10) IS
    SELECT * FROM pracownicy WHERE id_dzialu=p_id;
  vr_pracownik pracownicy%ROWTYPE;
  ...
BEGIN
  OPEN c_dzial(20);
  FETCH c_dzial INTO v_pracownik;
  ...
END;
```

Zamknięcie kursora


```
CLOSE nazwa;
```

Np.:

```
DECLARE
  CURSOR c_dzial (p_id NUMBER := 10) IS
    SELECT * FROM pracownicy WHERE id_dzialu=p_id;
  vr_pracownik pracownicy%ROWTYPE;
  ...
BEGIN
  OPEN c_dzial(20);
  FETCH c_dzial INTO v_pracownik;
  CLOSE c_dzial;
  ...
END;
```

Atrybuty kursora

Każdy kursor posiada atrybuty:

- `%ISOPEN` - zwraca prawdę jeżeli kursor jest otwarty (dla niejawnego zawsze `FALSE`),
- `%FOUND` - zwraca prawdę jeżeli ostatnie pobranie (`FETCH`) lub modyfikacja danych powiodła się,
- `%NOTFOUND` - zwraca prawdę jeżeli ostatnie pobranie lub modyfikacja danych nie powiodła się,
- `%ROWCOUNT` - zwraca liczbę wierszy pobranych lub zmodyfikowanych od momentu otwarcia kursora.

Pętle obsługujące kursor I

```
DECLARE
  CURSOR c_dzial (p_id NUMBER := 10) IS
    SELECT * FROM pracownicy WHERE id_dzialu=p_id;
  vr_pracownik pracownicy%ROWTYPE;
BEGIN
  OPEN c_dzial(20);
  LOOP
    FETCH c_dzial INTO vr_pracownik;
    IF c_dzial%NOTFOUND THEN
      EXIT;
    END IF;
    dbms_output.put_line(vr_pracownik.nazwisko);
  END LOOP;
  dbms_output.put_line('Wybrano '||
    c_dzial%ROWCOUNT||' wierszy.');
  CLOSE c_dzial;
END;
```

Pętle obsługujące kursor II


```
DECLARE
  CURSOR c_dzial (p_id NUMBER := 10) IS
    SELECT * FROM pracownicy WHERE id_dzialu=p_id;
BEGIN
  FOR vr_pracownik IN c_dzial(20) LOOP
    dbms_output.put_line(vr_pracownik.nazwisko);
  END LOOP;
END;
```

Pętle obsługujące kursor III


```
DECLARE
  v_id pracownicy.id_dzialu%TYPE := &id;
BEGIN
  FOR vr_pracownik IN (SELECT * FROM pracownicy
   WHERE id_dzialu = v_id ) LOOP
    dbms_output.put_line(vr_pracownik.nazwisko);
  END LOOP;
END;
```

Wyjątek

Wyjątek

- zdarzenie (błąd lub ostrzeżenie), które może wystąpić w czasie wykonywania bloku PL/SQL. Zdarzenie to powoduje przerwanie normalnego toku działania programu i przeniesienie do sekcji EXCEPTION, w celu wykonania instrukcji przewidzianych do obsługi danej sytuacji. Wyróżniamy wyjątki predefiniowane oraz użytkownika.

Wyjątki predefiniowane I

Niektóre wyjątki zgłaszane automatycznie przez system:

- `CASE_NOT_FOUND` - żadna z podanych klazul `WHEN` nie spełnia warunku, a nie przewidziano sekcji domyślnej `ELSE`.
- `CURSOR_ALREADY_OPEN` - próba otwarcia otwartego kursora.
- `DUP_VAL_ON_INDEX` - próba naruszenia ograniczenia unikalnej wartości.
- `INVALID_CURSOR` - nielegalna operacja na kursorze.
- `INVALID_NUMBER` - błąd wartości.
- `LOGIN_DENIED` - nieudana próba połączenia do bazy.
- `NO_DATA_FOUND` - instrukcja `SELECT` nie zwróciła żadnego wiersza.
- `NOT_LOGGED_ON` - nie połączono się z bazą.

Wyjątki predefiniowane II

- `TIMEOUT_ON_RESOURCE` - przekroczenie czasu oczekiwania na zwolnienie zasobu.
- `TOO_MANY_ROWS` - instrukcja `SELECT` zwróciła więcej niż jeden wiersz.
- `VALUE_ERROR` - błąd wartości.
- `ZERO_DIVIDE` - próba dzielenia przez zero.

Identyfikacja błędu:

- `SQLERRM` - opis wyjątku,
- `SQLCODE` - numer wyjątku.

Korzystanie z wyjątków użytkownika I

- deklaracja

```
nazwa_wyjatku EXCEPTION;
```

- podniesienie wyjątku

```
RAISE nazwa_wyjatku;
```

- przechwycenie w sekcji obsługi wyjątków.

```
EXCEPTION
  WHEN wyjatek1 THEN
    instrukcje1;
  WHEN wyjatek2 THEN
    instrukcje2;
  ...
  WHEN OTHERS THEN
    instrukcje_n;
```

Korzystanie z wyjątków użytkownika II

```
DECLARE
  e_zla_wartosc EXCEPTION;
  ...
BEGIN
  ...
  IF v_id = 0 THEN
    RAISE e_zla_wartosc;
  END IF;
  ...
  EXCEPTION
    WHEN e_zla_wartosc THEN
      dbms_output.put_line('Bledna wartosc
          identyfikatora.');
END;
```

Propagacja wyjątków I

```
BEGIN
  ...
  BEGIN
    ...
    IF v_id = 0 THEN
      RAISE e_zla_wartosc;
    END IF;
    ...
    EXCEPTION
      WHEN e_zla_wartosc THEN
        dbms_output.put_line('Bledna wartosc
          identyfikatora.');
  END;
  ...
  EXCEPTION
    ...
END;
```

Propagacja wyjątków II

```
BEGIN
  ...
  BEGIN
    ...
    IF v_id = 0 THEN
      RAISE e_zla_wartosc;
    END IF;
    ...
    EXCEPTION
    WHEN TOO_MANY_ROWS THEN
      dbms_output.put_line('Za duzo wartosci.');
  END;
  ...
  EXCEPTION
    WHEN e_zla_wartosc THEN
      dbms_output.put_line('Bledna wartosc ident...');
END;
```

Propagacja wyjątków III

```
BEGIN
  ...
  BEGIN
    ...
    IF v_id = 0 THEN
      RAISE e_zla_wartosc;
    END IF;
    ...
    EXCEPTION
    WHEN TOO_MANY_ROWS THEN
      dbms_output.put_line('Za duzo wartosci.');
  END;
  ...
  EXCEPTION
    WHEN VALUE_ERROR THEN
      dbms_output.put_line('Blad konwersji.');
END;
```

Raise_application_error

Wywołania procedury `Raise_application_error` pozwala przerwać działanie programu i wyprowadzić na ekran informacje o błędzie.
Składnia:

```
Raise_application_error(numer_bledu, opis);
```

Numer błedu zawiera się w przedziale od -20000 do -20999, a opis może zajmować 2048 bajtów. Np.:

```
IF v_id = 0 THEN
  Raise_application_error(-20002,
    'Błędny identyfikator');
ELSE
  ...
END IF;
```

Podsumowanie

PL/SQL umożliwia:

- korzystanie ze zmiennych, stałych,
- instrukcji sterujących,
- przetwarzanie sekwencyjne zorientowane na dane,
- obsługę sytuacji wyjątkowych,
- i wiele innych, o czym na następnym wykładzie...

Źródła

W wykładzie wykorzystano materiały:

- `http://www.oracle.com/technology/tech/pl_sql/index.html`
- M. Lentner, Oracle 9i Kompletny podręcznik użytkownika, PJWSTK - W-wa, 2003
- `http://wazniak.mimuw.edu.pl/index.php?title=Bazy_danych`
- `http://www.ploug.org.pl/showhtml.php?file=szkola/szkola_9/materialy`
- Garcia-Molina, Ullman, Widom: Implementacja systemów baz danych, WNT 2003