Wykład III

Braki w SQL

- obsługi zdarzeń i sytuacji wyjątkowych
- funkcji i procedur użytkownika
- definiowania złożonych ograniczeń integralnościowych

Proceduralny SQL

Transact- SQL *używany przez Microsoft SQL Server*
PL/pgSQL- *zaimplementowany przez PostgreSQL*

PL/SQL – (*ang. Procedural Language/Structured Query Language*)
Proceduralne rozszerzenie języka SQL do pisania aplikacji ***w Oracle***

- język proceduralny; kompilowany
- wykorzystuje typowe konstrukcje programistyczne; pętle, decyzje
- ma strukturę blokową
- każda instrukcja bloku PL/SQL jest zakończona średnikiem
- operator przypisania to **:=**
- bloki mogą być zagnieżdżone
- jedynymi poleceniami języka SQL, które mogą pojawić się w bloku PL/SQL są:
- **INSERT, UPDATE, DELETE , SELECT (** *zmiany w składni* **)** oraz obsługi transakcji
- obsługuje wszystkie typy SQL , oraz swoje dodatkowe typy np. BOOLEAN, typy złożone; RECORD, TABLE, VARRAY oraz typy definiowane przez użytkownika

Blok anonimowy

[DECLARE <*deklaracje obiektów; zmienne, stałe, wyjątki,…>***]**
BEGIN
<*ciąg instrukcji do wykonania*> **;**
[EXCEPTION <*obsługa wyjątków>;* **]**
END ;

Deklaracja zmiennych

DECLARE
<*nazwa_zmiennej*> **typ** [(<*rozmiar*>)] **[NOT NULL] [{ := | DEFAULT }** <*wartość*> **];**

przykłady deklaracji

znak **CHAR(1);**
wynagrodzenie **NUMBER(7,2);**
pi CONSTANT NUMBER(7,5) := 3.14159 ;
nazwa **VARCHAR2(10) :=** ‘DRUKARKA’ ;
termin **DATE :=** Sysdate ;

stan_cywilny **BOOLEAN :=** FALSE ;
brak_danych **EXCEPTION ;**
nazwisko_osoby podatnicy.nazwisko**%TYPE** ;
osoba_rekord podatnicy%**ROWTYPE** ;

Instrukcje sterujące

IF <*warunek_logiczny*> **THEN**
<*blok instrukcji* **;>**
END IF ;

IF <*warunek_logiczny*> **THEN**
< *blok_1 instrukcji* **;>**
ELSE <*blok_2 instrukcji*> **END IF ;**

IF <*warunek_logiczny_1*> **THEN**
<*blok_1 instrukcji* **;>**
ELSIF <*warunek_logiczny_2*> **THEN**
<*blok_2 instrukcji* **;>**
END IF ;

CASE
WHEN <*warunek_logiczny_1*> **THEN** <*blok_1 instrukcji* **;>**
WHEN <*warunek_logiczny_2*> **THEN** <*blok_2 instrukcji* **;>**
. . .
[ELSE <*blok_n instrukcji*> **;]**
END CASE ;

LOOP
<*blok instrukcji* >**;**
(w tym **EXIT** *lub* **EXIT WHEN** <*warunek logiczny*>*;)*
END LOOP ;

FOR <*zmienna*> **IN** <*wartość_1*> **..** <*wartość_2*>
LOOP
< *blok instrukcji* **;>**
END LOOP ;

WHILE <*warunek logiczny*>
LOOP
<*blok instrukcji* **;>**
END LOOP ;

Polecenia SQL w PL/SQL

wewnątrz bloków postać poleceń **INSERT, UPDATE, DELETE** pozostaje niezmieniona, dochodzi opcjonalna klauzula **RETURNING** <*wyrazenie*> **INTO** <*zmienna*>

przykłady:

```
SET SERVEROUTPUT ON
BEGIN
    LOOP
       EXIT WHEN (to_char(sysdate,'ss')='25');
    END LOOP;
    DBMS_OUT.put_line('Nadeszla 25 sekunda');
END;
```

```
CREATE TABLE Tymczasem
(Kol VARCHAR2(12));

DECLARE
v_day VARCHAR2(15);
BEGIN
FOR i IN 2014..2020 LOOP
 FOR j IN 01..12 LOOP
  v_day := TO_CHAR(TO_DATE('13-' ||j || '-' || i, 'DD-MM-YYYY'), 'DAY');
  IF (v_day LIKE 'PIĄTEK%') THEN
   INSERT INTO Tymczasem VALUES ('13-' || j || '-' || i);
  END IF;
 END LOOP;
END LOOP;
DBMS_OUTPUT.put_line('KONIEC');
END;
```

Specyficzną formę przyjmuje polecenie **SELECT** w przypadku, gdy wynikiem polecenia będzie **dokładnie jeden wiersz.**

SELECT *<lista_wynikowa>* **INTO** *<lista_zmiennych>* **FROM** *<źródło_danych>* . . . ;

SELECT *<lista_wynikowa>* **INTO** *<zmienna_rekordowa>* **FROM** *<źródło_danych>* . . . ;

Kursor

***n*azwany *o*bszar roboczy**, pozwalający składować i udostępniać wynik polecenia SQL wyróżnia się

- kursory jawne (deklarowane przez programistę)
- niejawne tworzone przez system o nazwie SQL

Obsługa kursora jawnego:

deklaracja

CURSOR *<nazwa_kursora>* [(*<parametr1>* <typ> [**:=** | **DEFAULT** *<wartość1>*]
[, *<parametr2>* . . .] . . .)]
[**RETURN** <typ>] **IS SELECT** . . . ;

Wykład III

otwarcie kursora

OPEN <*nazwa_kursora*> [(<*lista_parametrów*>)];

pobranie danych

FETCH <*nazwa_kursora*> **INTO** {< l*ista_zmiennych* > | < *zmienna_rekordowa* > };

zamknięcie kursora

CLOSE < *nazwa_kursora*>;

Atrybuty kursora:

%**ROWCOUNT** – zwraca liczbę wierszy pobranych lub zmodyfikowanych od momentu otwarcia kursora
%**FOUND** – zwraca prawdę jeśli ostatnie pobranie lub modyfikacja danych powiodła się
%**NOT_FOUND** – zwraca prawdę jeśli ostatnie pobranie lub modyfikacja danych nie powiodła się
%**ISOPEN** – zwraca prawdę jeśli kursor jest otwarty (dla niejawnego zawsze FALSE)

Przykłady:

```
DECLARE
  CURSOR c_zespol (p_id_zesp NUMBER :=20) IS
     SELECT * FROM Pracownicy WHERE id_zesp = p_id_zesp;
  vr_pracownik pracownicy%ROWTYPE;
BEGIN
  OPEN c_zespol(30);
  LOOP
    FETCH c_zespol INTO vr_pracownik;
    IF c_zespol%NOTFOUND  THEN
      EXIT;
    END IF;
    dbms_output.put_line(vr_pracownik.nazwisko||'  '||vr_pracownik.etat);
  END LOOP;
  dbms_output.put_line(' Wybrano '||c_zespol%ROWCOUNT|| ' wierszy ');
  CLOSE c_zespol;
END;
```

```
BEGIN
  FOR i IN ( SELECT nazwisko, etat FROM  Pracownicy)
  LOOP
    dbms_output.put_line(i.nazwisko||'  '||i.etat);
  END LOOP;
END;
```

Wyjątek

- zdarzenie (błąd lub ostrzeżenie), które może wystąpić w czasie wykonywania bloku PL/SQL.
- kiedy wyjątek jest podnoszony, następuje przerwanie normalnego toku działania programu i przeniesienie do sekcji EXCEPTION, w celu wykonania instrukcji przewidzianych do obsługi danej sytuacji.
- istnieje grupa wyjątków predefiniowanych oraz wyjątki użytkownika

Niektóre wyjątki zgłaszane autonomicznie przez system:

CASE_NOT_FOUND – żadna z podanych klauzul WHEN nie spełnia warunku, a nie przewidziano sekcji domyślnej ELSE
NO_DATA_FOUND – instrukcja SELECT nie zwróciła żadnego wiersza
TOO_MANY_ROWS – instrukcja SELECT zwróciła więcej niż jeden wierszowych
INVALID_NUMBER – nieudana konwersja na liczbę
VALUE_ERROR – błąd obcięcia, artymetyczny
ZERO_DIVIDE – próba dzielenia przez zero
CURSOR_ALREADY_OPEN – próba otwarcia otwartego kursora
INVALID_CURSOR – nielegalna operacja na kursorze

Identyfikacja błędu:

SQLERRM – opis wyjątku
SQLCODE – numer wyjątku

Korzystanie z wyjątków użytkownika

deklaracja

<*nazwa_wyjątku*> **EXCEPTION**;

podniesienie wyjątku

RAISE <*nazwa_wyjątku*>;

przechwycenie w sekcji obsługi wyjątków

```
EXCEPTION
    WHEN <wyjątek1> THEN
        <instrukcje1;>
    WHEN <wyjątek2> THEN
        <instrukcje2;>
     . . .
    WHEN OTHERS THEN
        <instrukcje_n;>
```

Propagacja wyjątków

```
BEGIN
  . . .
 BEGIN
    . . .
   IF v_id =0 THEN
     RAISE e_zla_wartosc;
    END  IF;
    . . .
    EXCEPTION
    WHEN  TOO_MANY_ROWS  THEN
        dbms_output.put_line (' za dużo wartości');
 END;
 . . .
EXCEPTION
  WHEN e_zla_wartosc  THEN
      dbms_output.put_line ('błędna wartość');
END;
```

RAISE_APPLICATION_ERROR – rodzaj procedury, która pozwala przerwać działanie programu i wyprowadzić na ekran informacje o błędzie

składnia

RAISE_APPLICATION_ERROR(<*numer_błędu*> , <*opis_błędu*>);

gdzie

numer błędu jest z przedziału od -20000 do -20999,
opis może zajmować co najwyżej 512 znaków

style stosowania wyjątków; kiedy używać RAISE_APPLICATION_ERROR zamiast RAISE