

## Rozdział 10a Kursory

Kursory i praca z kursorami, otwieranie kursora, pobieranie z kursora, zamykanie kursora, zmienne cursorowe, wyrażenie CURSOR, kursory niejawne



## Kursor jawny

Każde zapytanie SQL umieszczone w programie PL/SQL może zwrócić zero, jedną bądź wiele krotek. Aby efektywnie przetworzyć krotki zwrócone przez zapytanie korzystamy z kursorów. Kursor jest obiektem związanym z zapytaniem. Programista musi:

- Zadeklarować kursor.
- Otworzyć kursor (zidentyfikować zbiór wynikowy).
- Pobrać daną do kursora (odczytać kolejną krotkę z wyniku zapytania i wpisać ją do kursora).
- Zamknąć kursor (zwolnić obszar pamięci przydzielony kursorowi).

Kursor to nazwa obszaru roboczego, w którym mieści się wynik zapytania (ang. *result set*). Wewnątrz kursora wyróżniamy bieżący wiersz (ang. *current row*).



## Deklarowanie kursora (1)

- Kursor bezparametrowy:

```
DECLARE CURSOR nazwa_kursora IS zapytanie_SQL;
```

- Nazwa kursora nie jest zmienną, lecz identyfikatorem. Do kursora nie można przypisać wartości.
- Przykład:

```
DECLARE CURSOR c_zespoly IS
  SELECT nazwa, adres
  FROM zespoly
  ORDER BY nazwa;
BEGIN
  ...
```



## Deklarowanie kursora (2)

- Kursor sparymetryzowany

```
DECLARE CURSOR nazwa_kursora(lista_parametrów_formalnych)
  IS zapytanie_SQL;
```

- Deklaracja parametru

```
parametr typ_wartości [ { := | DEFAULT } wartość ]
```

- Parametry są widoczne tylko wewnątrz kursora, nie można związać z nimi żadnych ograniczeń.
- Nie podajemy długości dla typu wartości parametru.

```
DECLARE CURSOR c_pracownicy (p_zespol NUMBER DEFAULT 10)
  IS SELECT * FROM pracownicy
  WHERE id_zesp = p_zespol ORDER BY nazwisko;
BEGIN
  ...
```



## Otwieranie kursora

Otwarcie kursora powoduje wykonanie związanego z nim zapytania i zidentyfikowanie zbioru wynikowego, zawierającego rekordy spełniające kryteria wyszukiwania.

```
OPEN nazwa_kursora [(lista_parametrów_aktualnych)];
```

```
DECLARE
  CURSOR c_zespoly IS
    SELECT nazwa, adres FROM zespoly ORDER BY nazwa;
  CURSOR c_pracownicy (p_zespol NUMBER DEFAULT 10) IS
    SELECT * FROM pracownicy
    WHERE id_zesp = p_zespol ORDER BY nazwisko;
BEGIN
  OPEN c_zespoly;
  ...
  OPEN c_pracownicy(20);
  ... -- Zamykamy kursor c_pracownicy
  OPEN c_pracownicy; -- Parametr P_ZESPOL przyjmuje wartość domyślną równą 10
  ...
```

## Pobieranie rekordów z kursora (1)

```
FETCH nazwa_kursora
INTO lista_zmiennych_prostych | zmienna_rekordowa;
```

Wykonanie polecenia FETCH powoduje odczytanie bieżącego wiersza kursora i przesunięcie znacznika kursora na kolejny wiersz. Na liście zmiennych musi się znajdować taka sama liczba zmiennych jak liczba atrybutów w kursorze. Odpowiednie zmienne i atrybuty muszą się zgadzać co do typu.

```
DECLARE
  CURSOR c_zespoly IS
    SELECT nazwa, adres FROM zespoly ORDER BY nazwa;
  v_nazwa zespoly.nazwa%TYPE;
  v_adres zespoly.adres%TYPE;
BEGIN
  OPEN c_zespoly;
  FETCH c_zespoly INTO v_nazwa, v_adres; -- użycie zmiennych prostych
  ...
```

## Pobieranie rekordów z kursora (2)

```
DECLARE
  CURSOR c_pracownicy (p_zespol NUMBER DEFAULT 10) IS
    SELECT * FROM pracownicy
    WHERE id_zesp = p_zespol ORDER BY nazwisko;
  r_pracownik pracownicy%ROWTYPE; -- definicja w oparciu o strukturę rekordu relacji
BEGIN
  OPEN c_pracownicy(40);
  FETCH c_pracownicy INTO r_pracownik; -- użycie zmiennej rekordowej
  ...
```

```
DECLARE
  CURSOR c_zespoly IS
    SELECT nazwa, adres FROM zespoly ORDER BY nazwa;
  r_c_zespoly c_zespoly%ROWTYPE; -- definicja w oparciu o strukturę rekordu kursora
BEGIN
  OPEN c_zespoly;
  FETCH c_zespoly INTO r_c_zespoly; -- użycie zmiennej rekordowej
  ...
```

## Zamykanie kursora

```
CLOSE nazwa_kursora;
```

Zamknięcie kursora powoduje, że kursor staje się nieaktywny a zbiór wynikowy związany z kursorem staje się niezdefiniowany. Zamknięty kursor można powtórnie otworzyć, np. z innymi parametrami. Każde odwołanie się do zamkniętego (lub jeszcze nie otwartego) kursora powoduje błąd **INVALID\_CURSOR**.

```
DECLARE
  CURSOR c_zespoly IS
    SELECT nazwa, adres FROM zespoly ORDER BY nazwa;
  v_nazwa zespoly.nazwa%TYPE;
  v_adres zespoly.adres%TYPE;
BEGIN
  OPEN c_zespoly;
  FETCH c_zespoly INTO v_nazwa, v_adres;
  CLOSE c_zespoly;
  ...
```

## Atrybuty kursora (1)

- **%FOUND** – wartością atrybutu jest **TRUE** jeśli ostatnia operacja FETCH odczytała krotkę z kursora. W przeciwnym wypadku (tzn. kiedy odczyt się nie udał) atrybut przyjmuje wartość **FALSE**. Przed pierwszym odczytem atrybut ma wartość **NULL**
- **%NOTFOUND** – wartością atrybutu jest **FALSE** jeśli ostatnia operacja FETCH odczytała krotkę z kursora. W przeciwnym wypadku (tzn. kiedy odczyt się nie udał) atrybut przyjmuje wartość **TRUE**. Przed pierwszym odczytem atrybut ma wartość **NULL**
- **%ROWCOUNT** – wartością atrybutu jest liczba odczytanych z kursora krotek. Przed pierwszym odczytem atrybut ma wartość 0
- **%ISOPEN** – wartością atrybutu jest **TRUE** jeśli kursor jest otwarty i **FALSE** jeśli kursor jest zamknięty.



## Atrybuty kursora (2)

```
DECLARE
CURSOR c_zespoly IS
    SELECT nazwa, adres FROM zespoly ORDER BY nazwa;
v_nazwa zespoly.nazwa%TYPE;
v_adres zespoly.adres%TYPE;
BEGIN
OPEN c_zespoly;
LOOP
    FETCH c_zespoly INTO v_nazwa, v_adres;
    EXIT WHEN c_zespoly%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Zespół nr ' || c_zespoly%ROWCOUNT);
    DBMS_OUTPUT.PUT_LINE(v_nazwa || ', adres: ' || v_adres);
END LOOP;
CLOSE c_zespoly;
END;
```



## Pętla FOR z kursorem

```
DECLARE
CURSOR c_pracownicy (p_zespoly NUMBER DEFAULT 10) IS
    SELECT * FROM pracownicy
    WHERE id_zesp = p_zespoly ORDER BY nazwisko;
BEGIN
FOR r_prac IN c_pracownicy(20) LOOP
    DBMS_OUTPUT.PUT_LINE(r_prac.nazwisko ||
        ' zarabia ' || r_prac.placa_pod || ' i pracuje jako ' || r_prac.etat);
END LOOP;
END;
```

- Zmienna sterująca pętlą jest deklarowana automatycznie jako zmienna typu *kursor%ROWTYPE*.
- Kursor jest otwierany automatycznie.
- W każdym przebiegu pętli jedna krotka jest pobierana z kursora i umieszczana w zmiennej sterującej pętlą.
- Po pobraniu ostatniej krotki kursor jest automatycznie zamykany.



## Pętla FOR z podzapytaniem

```
BEGIN
FOR r_prac IN (SELECT * FROM pracownicy WHERE id_zesp = 20
    ORDER BY nazwisko) LOOP
    DBMS_OUTPUT.PUT_LINE(r_prac.nazwisko ||
        ' zarabia ' || r_prac.placa_pod || ' i pracuje jako ' || r_prac.etat);
END LOOP;
END;
```

- Zmienna sterująca pętlą jest deklarowana automatycznie jako zmienna typu *podzapytanie%ROWTYPE*.
- Podzapytanie może być parametryzowane przez użycie zmiennych w tekście zapytania.

```
DECLARE
v_id_zesp zespoly.id_zesp%TYPE;
BEGIN
v_id_zesp := 30;
FOR r_prac IN (SELECT * FROM pracownicy WHERE id_zesp = v_id_zesp
    ORDER BY nazwisko) LOOP
...

```



## Klauzula WHERE CURRENT OF (1)

- Ma zastosowanie do poleceń UPDATE i DELETE, kierowanych do zbioru rekordów kursora. Warunek jest spełniony tylko i wyłącznie dla bieżącego rekordu w kursorze.
- Zapytanie definiujące kursor musi zawierać klauzulę FOR UPDATE OF (założenie blokady na odczytywanych krotkach).

```
DECLARE
CURSOR c_pracownicy (p_zespol NUMBER DEFAULT 10) IS
  SELECT * FROM pracownicy
  WHERE id_zesp = p_zespol ORDER BY nazwisko FOR UPDATE;
BEGIN
FOR r_prac IN c_pracownicy(10) LOOP
  IF (r_prac.etat = 'DYREKTOR') THEN
    UPDATE pracownicy SET placa_pod = 1.1 * placa_pod
    WHERE CURRENT OF c_pracownicy;
  ELSE
    UPDATE pracownicy SET placa_pod = 0.9 * placa_pod
    WHERE CURRENT OF c_pracownicy;
  END IF;
END LOOP;
END;
```

(c)

13

## Klauzula WHERE CURRENT OF (2)

- Jeśli zapytanie definiujące kursor używa połączenia, można w klauzuli FOR UPDATE wskazać relację, do której będą kierowane operacje DML.
- Relację wskazuje się przez podanie dowolnego atrybutu z relacji.

```
DECLARE
CURSOR c_prac_zesp IS
  SELECT nazwisko, placa_pod, nazwa
  FROM pracownicy JOIN zespolny USING(id_zesp)
  FOR UPDATE OF nazwisko;
BEGIN
FOR r_pz IN c_prac_zesp LOOP
  IF (r_pz.nazwa = 'ADMINISTRACJA') THEN
    DELETE pracownicy WHERE CURRENT OF c_prac_zesp;
  ELSE
    UPDATE pracownicy SET placa_pod = 2 * placa_pod
    WHERE CURRENT OF c_prac_zesp;
  END IF;
END LOOP;
END;
```

(c) Instytut Informatyki Politechniki Poznańskiej

14

## Zmienna kursorowa

- Jest referencją do obiektu reprezentującego kursor lub zapytanie, nie jest kursorem!
- Pozwala na użycie kursora udostępniającego różne zbiory rekordów.
- Rodzaje zmiennych kursorowych:
  - z określoną strukturą rekordu – zmienna kursorowa silnie typowana,
  - bez określonej struktury rekordu – zmienna kursorowa słabo typowana.
- Proces deklarowanie zmiennej kursorowej:
  1. zadeklarowanie typu REF CURSOR,

```
TYPE nazwa_typu IS REF CURSOR [RETURN definicja_rekordu];
```

2. zadeklarowanie właściwej zmiennej za pomocą typu z p. 1.

```
nazwa_zmiennej_kursorowej nazwa_typu;
```

(c) Instytut Informatyki Politechniki Poznańskiej

15

## Zmienna kursorowa silnie typowana

```
DECLARE
TYPE tPracownik IS REF CURSOR RETURN pracownicy%ROWTYPE;
v_zmienna_kur tPracownik;
r_pracownik pracownicy%ROWTYPE;
BEGIN
OPEN v_zmienna_kur FOR SELECT * FROM pracownicy
  WHERE etat = 'PROFESOR';
LOOP
  FETCH v_zmienna_kur INTO r_pracownik;
  EXIT WHEN v_zmienna_kur %NOTFOUND;
  DBMS_OUTPUT.PUT_LINE (r_pracownik.nazwisko);
END LOOP;
CLOSE v_zmienna_kur;
OPEN v_zmienna_kur FOR SELECT * FROM pracownicy
  WHERE etat = 'ADIUNKT';
LOOP
  ...
END LOOP;
CLOSE v_zmienna_kur;
END;
```

(c) Instytut Informatyki Politechniki Poznańskiej

16

## Zmienna kursorowa słabo typowana

```
DECLARE
TYPE tKursor IS REF CURSOR;
v_zmienna_kur tKursor;
r_pracownik pracownicy%ROWTYPE; r_zespol zespol%ROWTYPE;
BEGIN
OPEN v_zmienna_kur FOR SELECT * FROM pracownicy;
LOOP
    FETCH v_zmienna_kur INTO r_pracownik;
    EXIT WHEN v_zmienna_kur %NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (r_pracownik.nazwisko);
END LOOP;
CLOSE v_zmienna_kur;
OPEN v_zmienna_kur FOR SELECT * FROM zespol;
LOOP
    FETCH v_zmienna_kur INTO r_zespol;
    EXIT WHEN v_zmienna_kur %NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (r_zespol.nazwa);
END LOOP;
CLOSE v_zmienna_kur;
END;
```

(c) Instytut Informatyki Politechniki Poznańskiej

17

## Wyrażenie CURSOR

- Pozwala na zdefiniowanie kursora zagnieżdżonego w zapytaniu.
- Używane zarówno w poleceniach SQL jak i programach PL/SQL.
- Składnia:

```
SELECT ..., CURSOR(zapytanie) FROM ...
```

- Zasady stosowania:
  - tylko dla kursorów jawnych,
  - kursor obsługujący zagnieżdżone zapytanie jest otwierany i zamykany automatycznie,
  - dopuszczalne wielopoziomowe zagnieżdżanie wyrażenia.

(c) Instytut Informatyki Politechniki Poznańskiej

18

## Wyrażenie CURSOR w poleceniu SQL

```
SQL> SELECT nazwa,
        CURSOR(SELECT nazwisko, placa_pod
                FROM pracownicy p
                WHERE id_zesp = z.id_zesp
                ORDER BY nazwisko) AS pracownik
        FROM zespol z ORDER BY nazwa;
```

```
NAZWA          PRACOWNIK
-----
ADMINISTRACJA  CURSOR STATEMENT : 2

CURSOR STATEMENT : 2
NAZWISKO      PLACA_POD
-----
MAREK         410,2
WEGLARZ       1730

ALGORYTMY     CURSOR STATEMENT : 2

CURSOR STATEMENT : 2
NAZWISKO      PLACA_POD
-----
BLAZEWICZ    1350

BADANIA OPERACYJNE  CURSOR STATEMENT : 2

CURSOR STATEMENT : 2
nie wybrano żadnych wierszy
...
```

(c) Instytut Informatyki Politechniki Poznańskiej

19

## Wyrażenie CURSOR w PL/SQL

```
DECLARE
TYPE tKursor IS REF CURSOR;
CURSOR cWszystko IS
    SELECT nazwa, CURSOR(SELECT nazwisko FROM pracownicy p
                        WHERE id_zesp = z.id_zesp ORDER BY nazwisko) AS pracownik
    FROM zespol z ORDER BY nazwa;
v_pracownicy tKursor;
v_nazwa zespol.nazwa%TYPE;
v_nazwisko pracownicy.nazwisko%TYPE;
BEGIN
OPEN cWszystko;
LOOP
    FETCH cWszystko INTO v_nazwa, v_pracownicy;
    EXIT WHEN cWszystko %NOTFOUND;
    DBMS_OUTPUT.PUT_LINE ('Zespól: '||v_nazwa);
    LOOP -- brak OPEN
        FETCH v_pracownicy INTO v_nazwisko;
        EXIT WHEN v_pracownicy%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_pracownicy%ROWCOUNT||' '||v_nazwisko);
    END LOOP; -- brak CLOSE
END LOOP;
CLOSE cWszystko;
END;
```

(c) Instytut Informatyki Politechniki Poznańskiej

20

## Kursor niejawnny (1)

- Każde polecenie DML (INSERT, UPDATE, DELETE, SELECT FOR UPDATE) powoduje utworzenie kursora niejawnnego (ang. *implicit cursor*).
- Atrybuty kursora niejawnnego pozwalają na sprawdzenie statusu ostatnio wykonanego polecenia DML:
  - **SQL%ROWCOUNT**: liczba wierszy zmodyfikowanych przez polecenie
  - **SQL%FOUND**: **TRUE** jeśli ostatnie polecenie zmodyfikowało jakiegokolwiek wiersze
  - **SQL%NOTFOUND**: **TRUE** jeśli ostatnie polecenie nie zmodyfikowało żadnych wierszy
  - **SQL%ISOPEN**: zawsze **FALSE** (kursor niejawnny jest zamykany natychmiast po zakończeniu polecenia)



## Kursor niejawnny (2)

```
BEGIN
INSERT INTO zespoly
  SELECT seq_zesp.NEXTVAL, nazwa||' (NOWY)', adres
FROM zespoly;
if SQL%FOUND then
  DBMS_OUTPUT.PUT_LINE ('Liczba nowych rekordów: '|| SQL%ROWCOUNT);
else
  DBMS_OUTPUT.PUT_LINE ('Nie wstawiono żadnego rekordu!');
end if;

DELETE pracownicy WHERE id_zesp in
  (SELECT id_zesp from zespoly
   WHERE nazwa = 'TESTOWY');
DBMS_OUTPUT.PUT_LINE('Liczba usuniętych rekordów: '|| SQL%ROWCOUNT);
END;
```

