Oracle PL/SQL

Paweł Rajba

pawel@ii.uni.wroc.pl
http://www.kursy24.eu/

Zawartość modułu 2

- Kusory
  - Wprowadzenie
  - Kursory użytkownika
  - Kursory domyślne
  - Zmienne kursora
  - Wyrażenia kursora

Wprowadzenie

- Co to jest kursor?
- Podział na kursory
  - użytkownika
  - domyślne
- Kiedy używać kursora?

Kursory użytkownika

- Definiowanie kursora
  - CURSOR nazwa IS zapytanie_select
- Używanie kursora
  - Otworzenie
  - Pobranie wyników do rekordu PL/SQL lub pojedynczych zmiennych
  - Zamknięcie kursora
- Powyższe można zrobić na 2 sposoby
  - OPEN, FETCH, CLOSE lub FOR LOOP

Kursory użytkownika

- Otwieranie kursora – uwagi
  - Po otwarciu zbiór wyników jest w tylko do odczytu
  - Wskaźnik kursora wskazuje na pierwszy wiersz
  - Otwarcie otwartego już kursora wygeneruje wyjątek CURSOR_ALREADY_OPEN
  - Zbiór wyników generowany jest w momencie otwarcie kursora, a nie jego deklaracji

Kursory użytkownika

- Pobieranie danych z kursora
  - bezpośrednio do zmiennych
    FETCH cursor_name INTO v1, v2, ..., vN;
  - do zmiennej rekordowej
    c_record cursor_name%ROWTYPE
    -- lub c_record table_name%ROWTYPE
    ...
    FETCH cursor_name INTO c_record
- Zamykanie kursora
  - CLOSE cursor_name;

Kursory użytkownika

- Pobieranie i zamykanie kusora – uwagi
  - Wczytanie z zamkniętego kursora spowoduje błąd
  - Zaleca się wczytywanie danych kursora do zmiennej rekordowej
  - Zaleca się zamykać każdy nieużywany już kursor, ponieważ istnieje ograniczenie ilości jednocześnie otwartych kursorów
    - tą liczbę można ustawić w konfiguracji bazy danych w pliku init.ora
  - Nie należy też zamykać już zamkniętego kursora

Kursory użytkownika

- Pobieranie wielokrotne z kursora
  - LOOP
    FETCH csr_name ...
    EXIT WHEN csr_name%NOTFOUND
    ... -- obsługa danych
    END LOOP
  - FETCH csr_name ...
    WHILE (csr_name%FOUND) LOOP
    ... -- obsługa danych
    FETCH csr_name ...
    END LOOP

Kursory użytkownika

- Pobieranie wielokrotne z kursora
  - FOR idx IN csr_name LOOP
    ... -- obsługa danych
    END LOOP
  - FOR idx IN (SELECT ... FROM ...) LOOP
    ... -- obsługa danych
    END LOOP

Kursory użytkownika

- Pobieranie wielokrotne z kursora – uwagi:
  - zmiennej idx nie deklarujemy – będzie automatycznie zadeklarowana z typem %ROWTYPE
  - przy przetwarzaniu wszystkich wierszy kursora zaleca się stosowanie składni FOR, szczególnie tej drugiej
- Jeszcze jedna uwaga:
  - jeśli w deklaracji kursora jest użyta zmienna, to wartość jest pobierana już w momencie otwierania a nie w momencie deklaracji

Kursory użytkownika

- Atrybuty
  - %FOUND
  - %ISOPEN
  - %NOTFOUND
  - %ROWCOUNT
- Przykład: kursory-petle.txt

Kursory użytkownika

- Kursory sparametryzowane
  - Kursor może przyjmować kilka parametrów
  - Bardzo przydatne przy kursorach zagnieżdżonych
- Deklaracja
  - CURSOR csr_name(par1, par2, ...) IS SELECT ...
- Podanie parametrów aktualnych
  - OPEN csr_name(v1, v2, ...) ...
  - FOR idx IN csr_name(v1, v2, ...) ...
- Przykład: kursory-parametry.txt

Kursory użytkownika

- Kursory SELECT FOR UPDATE
  - Pozwalają na modyfikację danych kursora
- Deklaracja
  - CURSOR csr_name IS SELECT c1, c2, ... FROM t FOR UPDATE [OF mcol1, mcol2, ...]
    - mcol1, mcol2, ... - kolumny, które będą mogły być modyfikowane
    - niepodanie klauzuli z kolumnami spowoduje możliwość modyfikacji dowolnej kolumny

Kursory użytkownika

- Kursory SELECT FOR UPDATE – używanie
  - Możemy wykonywać operacje UPDATE, DELETE
  - Rekord do modyfikacji określamy poprzez klauzulę WHERE CURRENT of csr_name
- Kursory SELECT FOR UPDATE – uwaga
  - przy otwarciu tego kursora nakładana jest blokada na rekordy, które się w danym kursorze znajdują
- Przykład: kursory-update.txt

Kursory domyślne

- Służą do informacji o przebiegu pojedynczych instrucji typu
  - insert, update, delete, select .. into
- Kursor domyślny nazywa się też SQL
- W momencie wykonania instrukcji, kursor jest otwierany, po zakończeniu – zamykany
  - W związku z tym nie tutaj poleceń OPEN, FETCH i CLOSE

Kursory domyślne

- Atrybuty
  - SQL%FOUND – wskazuje, czy ostatnio wykonana instrukcja wpłynęła na jakiś wiersz, czy nie
  - SQL%ISOPEN – zawsze ustawiony na false
  - SQL%NOTFOUND – analogiczne do SQL%FOUND
  - SQL%ROWCOUNT – ilość zmodyfikowanych wierszy
- Przykład: kursory-domyslne.txt

Zmienne kursora

- Pozwala na kojarzenie zmiennej z więcej niż jednym zapytaniem
- Zmienna ta jest wskaźnikiem
- Pozwala na dynamiczne tworzenie zapytań zależne od przebiegu kodu PL/SQL

Zmienne kursora

- Deklaracja zmiennej kursora
  - Słaby typ
    - TYPE rc IS REF CURSOR;
      v_rc rc;
  - Mocny typ
    - TYPE rc IS REF CURSOR tab%ROW_TYPE;
      v_rc rc;
    - TYPE c_rec IS RECORD (Id INT, Name VARCHAR2(20));
      TYPE rc IS REF CURSOR
      v_rc rc;

Zmienne kursora

- Użycie zmiennej kursora
  - deklaracja
  - otwarcie
    - OPEN v_rc FOR SELECT ...
  - pobieranie wierszy wyniku
  - zamknięcie
- Przypisywanie zmiennych
  - Słaby typ – dowolne przypisania
  - Mocny typ – typy muszą się zgadzać

Zmienne kursora

- Typ SYS_REFCURSOR
  - pozwala tworzyć zmienne słabego typu
  - deklaracja takiej zmiennej
    - v_rc SYS_REFCURSOR;
  - powyższe jest równoważne
    - TYPE rc IS REF CURSOR;
      v_rc rc;

Zmienne kursora

- Uwagi
  - korzystając ze zmiennych nie możemy korzystać z przekazywania parametrów do kursora
  - do zmiennych nie można przypisać wartości null
  - nie można porównywać zmiennych kursorowych oraz sprawdzać, czy mają wartość null
  - Atrybuty są takie jak w ogólnym przypadku kursorów
  - Zmiennych nie można składować w bazie danych
    - czyli zmiennych o typach ref cursor i sys_refcursor
- Przykład: kursory-zmienne.txt

Wyrażenia kursora

- Pozwalają zmniejszyć liczbę deklaracji kursorów
- Mają większą wydajność niż konstrukcja zagnieżdżonych kursorów
- Mogą być zagnieżdżane dowolną liczbę razy
- Mogą być przekazywane do funkcji jako parametry (omówimy przy okazji funkcji)
- Deklaracja
  - cursor csr is select col1, cursor (select ...) ...
- Przykład: kursory-wyrazenia.txt

