

Границы моего языка означают границы
моего мира.

Людвиг Виттгенштейн

ОСНОВЫ ЯЗЫКА PL/SQL

Взаимодействие с Oracle

Взаимодействие с Oracle

Что можно использовать в PL/SQL:

- 1) Команды INSERT, UPDATE, DELETE, SELECT и LOCK TABLE.
- 2) Команды управления транзакциями (COMMIT, ROLBACK, SAVE POINT и SET TRANSACTION - устанавливает режим транзакции "только чтение").
- 3) Все функции SQL, кроме агрегирующих.

```
IF a LIKE 'шаблон' THEN ...  
END IF;
```

- 4) Псевдостолбцы:

```
CURRVAL  
LEVEL  
NEXTVAL  
ROWID  
ROWNUM
```

```
INSERT INTO emp VALUES (empno_seq.NEXTVAL, new_ename, ...);
```

Псевдостолбцы

NEXTVAL

Возвращает следующее значение указанной последовательности.
имя_последовательности.NEXTVAL

CURRVAL

Возвращает текущее значение указанной последовательности.
имя_последовательности.CURRVAL

CURRVAL и NEXTVAL можно вызывать только в списке SELECT, фразе VALUES и фразе SET.

LEVEL

В предложении CONNECT BY команды SELECT псевдостолбец LEVEL позволяет организовать строки таблицы базы данных в иерархическую структуру. LEVEL возвращает номер уровня узла в этой структуре.

ROWID

Возвращает идентификатор строки (двоичный адрес строки) в таблице базы данных.

ROWNUM

Возвращает порядковый номер строки согласно тому порядку, в котором строки выбираются из таблицы.

```
SELECT * FROM emp WHERE ROWNUM < 10;
```

Операторы сравнения

ALL

Сравнивает значение с каждым значением в списке или в множестве, возвращаемом подзапросом, и возвращает TRUE, если все индивидуальные сравнения дают результат TRUE.

ANY, SOME

Сравнивает значение с каждым значением в списке или в множестве, возвращаемом подзапросом, и возвращает TRUE, если хотя бы одно из индивидуальных сравнений дает результат TRUE.

BETWEEN

Проверяет, лежит ли значение в интервале.

EXISTS

Возвращает TRUE, если подзапрос возвращает хотя бы одну строку.

IN

Проверяет на членство в множестве.

IS NULL

Проверяет на пустоту.

LIKE

Проверяет, удовлетворяет ли символьная строка заданному образцу, который может содержать поисковые символы.

Другие операторы

Операторы множеств: INTERSECT, MINUS, UNION и UNION ALL.

- INTERSECT возвращает все различающиеся строки, возвращенные обоими запросами.
- MINUS возвращает все различающиеся строки, возвращенные первым, но не вторым запросом.
- UNION возвращает все различающиеся строки, возвращенные любым из запросов.
- UNION ALL возвращает все строки, возвращенные любым из запросов, в том числе все повторяющиеся строки.

Все операторы строк, включая ALL, DISTINCT и PRIOR.

- ALL возвращает повторяющиеся строки в результате запроса или в агрегатном выражении.
- DISTINCT устраняет повторяющиеся строки в результате запроса или в агрегатном выражении.
- PRIOR ссылается на родительскую строку текущей строки, возвращенной иерархическим запросом.

Можно использовать советы оптимизатору.

Курсоры

Курсор – область памяти, в которую возвращается результат SQL-запроса.

Курсоры бывают:

- неявные: соответствуют командам DML (insert, update, delete), а также команде select, возвращающей одну строку;
- явные: описываются с помощью ключевого слова cursor.

Для объявления явного курсора используется синтаксис:

```
CURSOR имя [ (параметр [, параметр, ...]) ] IS  
SELECT ...;
```

"параметр", в свою очередь, имеет следующий синтаксис:

```
имя_переменной [IN] тип_данных [{:= | DEFAULT} значение]
```

Формальные параметры курсора должны иметь моду IN.

Примеры курсоров

Неявный:

```
DECLARE num number;  
begin  
    SELECT count(*) into num  
    FROM emp;  
...
```

Явный:

```
DECLARE  
    CURSOR c1 IS  
    SELECT ename, deptno FROM emp WHERE sal > 2000;  
...
```

Имя курсора – это необъявленный ранее идентификатор, а не переменная PL/SQL; его можно использовать только для обращения к запросу.

Работа с явным курсором

Открытие курсора:

```
OPEN имя_курсора[(параметры)];
```

Извлечение данных:

```
FETCH имя_курсора INTO переменная1 [, переменная2,...];
```

Закрытие курсора:

```
CLOSE имя_курсора;
```

DECLARE

```
my_sal emp.sal%TYPE;
```

```
my_job emp.job%TYPE;
```

```
factor INTEGER := 2;
```

```
CURSOR c1 IS
```

```
    SELECT factor*sal FROM emp WHERE job = my_job;
```

BEGIN ...

```
    OPEN c1;
```

```
    LOOP FETCH c1 INTO my_sal;
```

```
        EXIT WHEN c1%NOTFOUND;
```

```
    ...
```

```
        factor := factor + 1; -- не окажет влияния на FETCH
```

```
    END LOOP;
```

```
END;
```


Атрибуты курсора

		%NOTFOUND	%FOUND	%ROWCOUNT	%ISOPEN
OPEN	Перед	*	*	*	FALSE
	После	NULL	NULL	0	TRUE
Первая FETCH	Перед	NULL	NULL	0	TRUE
	После	FALSE	TRUE	1	TRUE
Промежуточные FETCH	Перед	FALSE	TRUE	1	TRUE
	После	FALSE	TRUE	**	TRUE
Последняя FETCH	Перед	FALSE	TRUE	**	TRUE
	После	TRUE	FALSE	**	TRUE
CLOSE	Перед	TRUE	FALSE	**	TRUE
	После	*	*	*	FALSE

* – возбуждает предопределенное исключение INVALID_CURSOR.

** – ЗАВИСИТ ОТ ДАННЫХ.

Неявный курсор (SQL)

Атрибуты: %NOTFOUND, %FOUND, %ROWCOUNT и %ISOPEN. Пока ORACLE автоматически не открыл курсор SQL, атрибуты неявного курсора возвращают NULL. Значения атрибутов неявного курсора всегда относятся к последней выполненной операции SQL.

```
UPDATE parts SET qty = qty - 1 WHERE partno = part_id;
sql_notfound := SQL%NOTFOUND;
check_parts;
IF sql_notfound THEN ...
END IF;
```

Другой пример:

```
DECLARE my_sal NUMBER(7,2);
        my_empno NUMBER(4);
BEGIN ...
SELECT sal INTO my_sal FROM emp WHERE
        empno = my_empno; -- может возбудить исключение
        NO_DATA_FOUND
IF SQL%NOTFOUND THEN -- получит управление при FALSE
... -- эти действия никогда не будут выполнены
END IF; ...
END;
```

Параметризованные курсоры

Курсору могут быть переданы параметры при открытии.

Пример:

```
DECLARE
```

```
    CURSOR c1 (my_ename CHAR, my_comm NUMBER,  
              edate date default trunc(sysdate))  
    IS SELECT ...
```

```
BEGIN
```

```
OPEN c1('ATTLEY', 300, '2012.10.12');
```

```
...
```

```
OPEN c1(employee_name, 150, to_char('20.09.2013', 'dd.mm.yyyy');
```

```
...
```

```
OPEN c1('THURSTON', my_comm);
```

```
...
```

Курсорные циклы FOR

Пример:

```
DECLARE
    result temp.col1%TYPE;
    CURSOR c1 IS SELECT n1, n2, n3 FROM data_table
        WHERE exper_num = 1;
BEGIN
    FOR c1rec IN c1 LOOP
        /* вычислить и сохранить результаты */
        result := c1rec.n2 / (c1rec.n1 + c1rec.n3);
        INSERT INTO temp VALUES (result, NULL, NULL);
    END LOOP;
    COMMIT;
END;
```

В одну строку:

```
FOR r1 IN (select * from emp) LOOP
    ...
END LOOP;
```

Параметризованный курсор в цикле FOR

```
DECLARE
    CURSOR emp_cursor(dnum NUMBER) IS
        SELECT sal, comm FROM emp WHERE deptno = dnum;
    total_wages NUMBER(11,2) := 0;
    high_paid NUMBER(4) := 0;
    higher_comm NUMBER(4) := 0;
BEGIN /*    Число итераций будет равно числу строк, *
*    возвращенных курсором emp_cursor.          */
FOR emp_record IN emp_cursor(20) LOOP
    emp_record.comm := NVL(emp_record.comm, 0);
    total_wages := total_wages + emp_record.sal + emp_record.comm;
    IF emp_record.sal > 2000.00 THEN high_paid := high_paid + 1;
    END IF;
    IF emp_record.comm > emp_record.sal
    THEN higher_comm := higher_comm + 1;
    END IF;
END LOOP;
INSERT INTO temp VALUES (high_paid, higher_comm, 'Total Wages: ' ||
    TO_CHAR(total_wages));
END;
```

Использование фразы FOR UPDATE

```
DECLARE
  CURSOR c1 IS
    SELECT ename, dname FROM emp, dept WHERE
      emp.deptno = dept.deptno AND job = 'MANAGER'
    FOR UPDATE OF sal;
-- блокирует emp, но не dept
```

```
DECLARE
  CURSOR c1 IS SELECT empno, job, sal FROM emp FOR UPDATE;
...
BEGIN
  OPEN c1;
  LOOP FETCH c1 INTO ...
  ...
  UPDATE emp SET sal = new_sal WHERE CURRENT OF c1;
  -- нельзя делать промежуточный commit !!!!!!!!!!!
  END LOOP; ...
END;
```