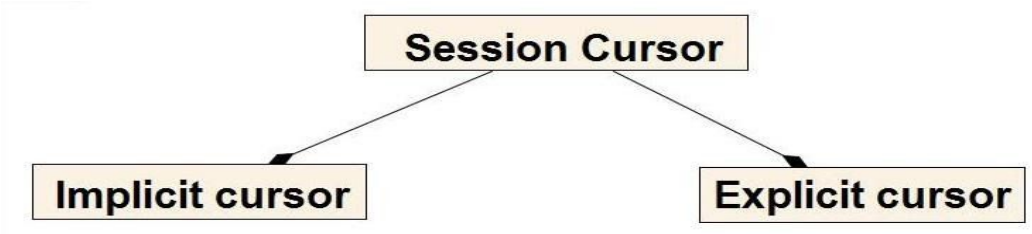
Оглавление

PL/SQL Cursors

Курсор – указатель на приватную SQL область, в которой содержится информация о выполнении конкретного select-оператора (или любого другого dml оператора).



Implicit Cursor

- ✓ PL/SQL открывает implicit cursor каждый раз, когда выполняет select или dml-оператор. Сразу после выполнения закрывает его.
- ✓ Мы не контролируем такие курсоры

✓ К некоторым атрибутам таких курсоров мы имеем доступ

Атрибут	Описание
SQL%ISOPEN	Всегда возвращает false
SQL%FOUND	Возвращает null, если никакого запроса не было выполнено; true, если запрос вернул какие-то записи (или повлиял на какие-то записи); иначе false
SQL%NOTFOUND	Возвращает null, если никакого запроса не было выполнено; false, если запрос вернул какие-то записи (или повлиял на какие-то записи); иначе true
SQL%ROWCOUNT	Количество записей, которые вернул запрос (на которые повлиял запрос)

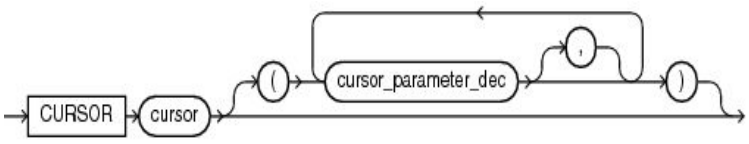
Пример:

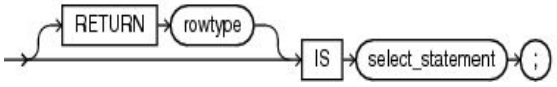
```
CREATE OR REPLACE PROCEDURE p(dept_no NUMBER) AUTHID DEFINER AS
BEGIN
    DELETE FROM dept_temp
    WHERE  department_id = dept_no;

    IF SQL%FOUND
    THEN
        DBMS_OUTPUT.PUT_LINE('Delete succeeded for department number ' ||
dept_no);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No department number ' || dept_no);
    END IF;
END;
```

Explicit cursor

✓ Это именованный курсор (named cursor).
✓ Объявление курсора

CURSOR *cursor_name* [*parameter_list*] [RETURN *return_type*] IS *select_statement*;





Restrictions:

- Select-оператор не может содержать конструкцию with

Пример:

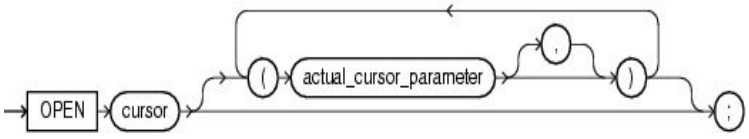
```
DECLARE
    CURSOR c1 IS
        SELECT employee_id,
               job_id,
               salary
        FROM   employees
        WHERE  salary > 2000;
BEGIN
    some_actions;
END;
```

Операции с курсором:

- Open cursor
- Close cursor
- Fetch from cursor

Open Cursor

1. Выделяет ресурсы базы данных для выполнения запроса
2. Выполняет запрос
3. Устанавливает позицию курсора перед первой записью результирующего набора

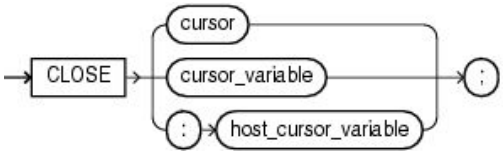


Возможные Exceptions

- INVALID_CURSOR
- CURSOR_ALREADY_OPEN

Close Cursor

Освобождает все ресурсы, связанные с данным курсором



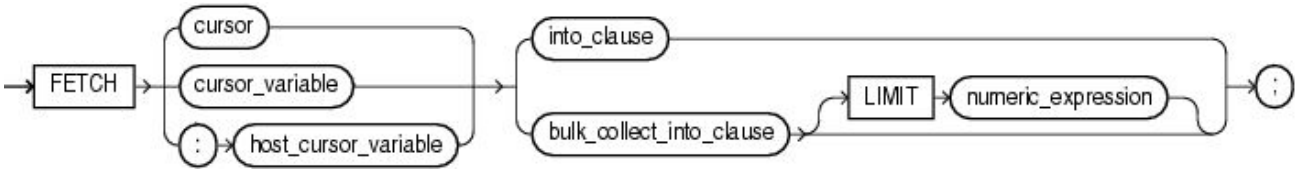
Возможные Exceptions

- INVALID_CURSOR
- CURSOR_ALREADY_OPEN

Fetch from cursor

FETCH *cursor_name* INTO *into_clause*

1. Извлекает текущую строку из курсора
2. Сохраняет значение колонок в переданных переменных
3. Переводит позицию курсора на следующую строку



- В операторе часто используются переменные, объявленные с помощью динамического типа %TYPE и %ROWTYPE
- Если в курсоре одно из полей – это некоторое выражение, для этого поля необходим псевдоним в двух случаях: если данные фетчатся в переменную типа %ROWTYPE, или если мы обращаемся к полю по имени

Пример (обратите внимание на использование внешних переменных в курсоре):

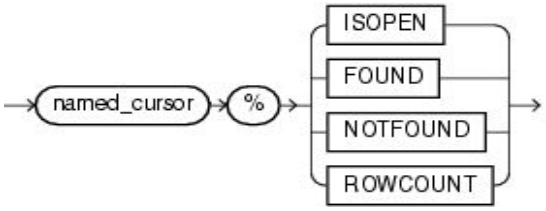
```
DECLARE
    sal          employees.salary%TYPE;
    sal_multiple employees.salary%TYPE;
    factor       INTEGER := 2;

    CURSOR c1 IS
        SELECT salary,
               salary * factor
        FROM   employees
        WHERE  job_id LIKE 'AD_%';

BEGIN
    OPEN c1;

    LOOP
        FETCH c1
            INTO sal, sal_multiple;
        EXIT WHEN c1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('factor = ' || factor);
        DBMS_OUTPUT.PUT_LINE('sal          = ' || sal);
        DBMS_OUTPUT.PUT_LINE('sal_multiple = ' || sal_multiple);
        factor := factor + 1;
    END LOOP;

    CLOSE c1;
END;
```

Курсор может принимать параметры:

```
DECLARE
    CURSOR c(job     VARCHAR2,
             max_sal NUMBER,
             hired   DATE DEFAULT '31-DEC-99') IS
        SELECT last_name,
               first_name,
               (salary - max_sal) overpayment
        FROM   employees
        WHERE  job_id = job
        AND    salary > max_sal
        AND    hire_date > hired
```

```
        ORDER  BY salary;

BEGIN
    OPEN c('SA_REP', 10000, '31-DEC-04');
    do_something;
    CLOSE c;
END;
```

Атрибуты курсора:



Атрибут	Описание
%ISOPEN	Возвращает true, если курсор открыт; иначе false. Хаще всего используется для проверки перед открытием или закрытием курсора
%FOUND	Возвращает null, если курсор открыт, но не было ни одного fetch; true, если последний fetch вернул записи; и false, если последний fetch не вернул ни одной записи
%NOTFOUND	Возвращает null, если курсор открыт, но не было ни одного fetch; false, если последний fetch вернул записи; и true, если последний fetch не вернул ни одной записи
%ROWCOUNT	Возвращает суммарное количество извлеченных записей

Для закрытого курсора обращение к любому атрибуту, кроме %ISOPEN, генерирует исключение **INVALID_CURSOR**

Пример:

```
DECLARE
    CURSOR c1 IS
        SELECT last_name,
               salary
        FROM   employees
        WHERE  ROWNUM < 11
        ORDER  BY last_name;

    my_ename  employees.last_name%TYPE;
    my_salary employees.salary%TYPE;
BEGIN
    OPEN c1;
    LOOP
```

```
        FETCH c1
            INTO my_ename,
                 my_salary;
        IF c1%NOTFOUND
        THEN
            EXIT;
        ELSE
            DBMS_OUTPUT.PUT_LINE('Name = ' || my_ename || ', salary = ' ||
my_salary);
        END IF;
    END LOOP;
END;
```

Select into

Получение одной записи:

```
SELECT select_item [, select_item ]...
 INTO variable_name [, variable_name ]...
 FROM table_name;
```

Получение нескольких записей:

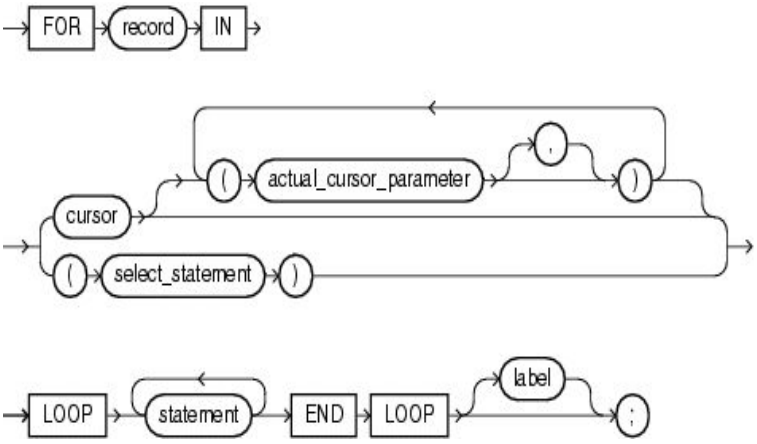
```
SELECT select_item [, select_item ]...
 BULK COLLECT INTO variable_name [, variable_name ]...
 FROM table_name;
```

Возможные Exceptions:

```
NO_DATA_FOUND
TOO_MANY_ROWS
```

Cursor for loop statement

- ✓ Используется implicit cursor (в этом случае называется implicit cursor FOR LOOP statement)
- ✓ На такой курсор нельзя ссылаться с помощью SQL
- ✓ В конструкции можно использовать explicit cursor (в этом случае называется explicit cursor FOR LOOP statement)
- ✓ В цикле неявно объявляется переменная типа cursor%ROWTYPE
- ✓ Переменная является локальной для цикла, доступна только в теле цикла и живет пока цикл выполняется
- ✓ Открывается курсор автоматически
- ✓ Закрывается также автоматически
- ✓ Курсор закрывается тогда, когда все записи выбраны или если выражение внутри цикла передает управление наружу цикла, либо же если возникает исключение
- ✓ Допустимы также курсоры с параметрами
- ✓ Каждый fetch выбирает неявно по 100 записей для улучшения производительности (начиная с 10 версии, до этого возвращалось по одной записи)



Пример:

```
DECLARE
    CURSOR c1 IS
        SELECT last_name,
               job_id
        FROM   employees
        WHERE  job_id LIKE '%CLERK%'
        AND    manager_id > 120
        ORDER  BY last_name;
BEGIN
    FOR item IN (SELECT last_name,
                        job_id
                 FROM   employees
                 WHERE  job_id LIKE '%CLERK%'
                 AND    manager_id > 120
                 ORDER  BY last_name)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Name = ' || item.last_name || ', Job = ' ||
item.job_id);
    END LOOP;

    FOR item IN c1
    LOOP
        DBMS_OUTPUT.PUT_LINE('Name = ' || item.last_name || ', Job = ' ||
item.job_id);
    END LOOP;
END;
```

Cursor variable (REF CURSOR)

Cursor variable – это указатель, т.е. содержит адрес объекта(курсора), а не сам объект.

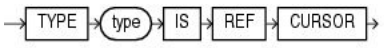
- ✓ Может использоваться для выполнения разных select-запросов
- ✓ Может участвовать в выражениях
- ✓ Может быть входным параметром

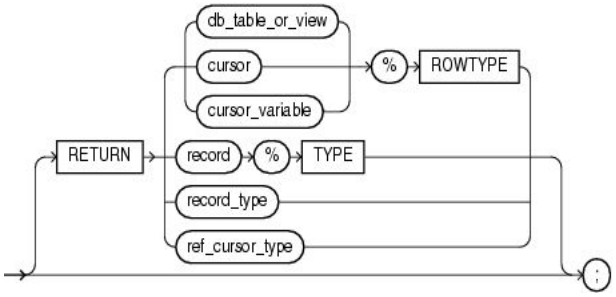
- ✓ Может быть параметром, передающимся от БД клиенту
- ✓ Не может содержать параметров
- ✓ **Сильный курсор** – если для него определен тип возвращаемого значения. К такому курсору можно привязывать только те запросы, которые возвращают набор данных определенной структуры.
- ✓ **Слабый курсор (SYS_REFCURSOR)** - курсор, для которого тип возвращаемого значения не определен. К нему можно привязывать любые запросы.

Операции с cursor variable

Объявление курсорной переменной

TYPE *type_name* IS REF CURSOR [RETURN *return_type*];

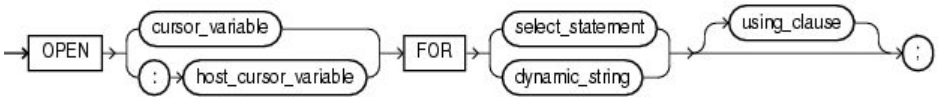




cursor_variable type_name;

Открытие и закрытие

OPEN cursor_variable FOR select_statement;



Открытие курсорной переменной выполняет те же действия, что и в случае explicit cursor. Разве что в начале связывает курсорную переменную с конкретным запросом (который может содержать bind переменные)

1. Выделяет ресурсы базы данных для выполнения запроса
2. Выполняет запрос
3. Устанавливает позицию курсора перед первой записью результирующего набора

CLOSE cursor_variable;

Необязательно закрывать курсор перед его переоткрытием с другим запросом. Если закрыть курсор,

его можно открыть заново (с последний связанным запросом).

Извлечение данных

FETCH

Связывание значения с курсорной переменной

target_cursor_variable := *source_cursor_variable*;

Исключения:

- ROWTYPE_MISMATCH

Атрибуты у курсорной переменной те же, что и у explicit cursor.

Пример:

```
DECLARE
    TYPE empcurtyp IS REF CURSOR RETURN employees%ROWTYPE; -- strong type
    TYPE genericcurtyp IS REF CURSOR; -- weak type

    cursor1   empcurtyp; -- strong cursor variable
    cursor2   genericcurtyp; -- weak cursor variable
    my_cursor SYS_REFCURSOR; -- weak cursor variable

    v_employees employees%ROWTYPE;

    TYPE emplist IS TABLE OF employees%ROWTYPE;
    emp_list emplist;

    v_sql varchar2(200) := 'SELECT * FROM   employees WHERE  REGEXP_LIKE(job_id,
''S[HT]_CLERK'') ORDER  BY last_name';
BEGIN
    OPEN cursor1 FOR v_sql;
    LOOP
        FETCH cursor1
            INTO v_employees;
        EXIT WHEN cursor1%NOTFOUND;
        some_actions;
    END LOOP;
    CLOSE cursor1;

    OPEN cursor1 FOR v_sql;
    FETCH cursor1 BULK COLLECT
        INTO emp_list;
    CLOSE cursor1;
    some_actions_with_emp_list;
END;
```

Cursor expression

✓ Возвращает вложенный курсор:

CURSOR (*subquery*)

- ✓ Открывается неявно при извлечении
- ✓ Закрывается либо явно пользователем, либо при закрытии родительского курсора

Пример:

```
DECLARE
  TYPE emp_cur_typ IS REF CURSOR;

  emp_cur    emp_cur_typ;
  dept_name  departments.department_name%TYPE;
  emp_name   employees.last_name%TYPE;

  CURSOR c1 IS
    SELECT department_name,
      CURSOR ( SELECT e.last_name
                 FROM employees e
                 WHERE e.department_id = d.department_id
                 ORDER BY e.last_name
               ) employees
    FROM departments d
    WHERE department_name LIKE 'A%'
    ORDER BY department_name;
BEGIN
  OPEN c1;
  LOOP  -- Process each row of query result set
    FETCH c1 INTO dept_name, emp_cur;
    EXIT WHEN c1%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Department: ' || dept_name);

    LOOP -- Process each row of subquery result set
      FETCH emp_cur INTO emp_name;
      EXIT WHEN emp_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('-- Employee: ' || emp_name);
    END LOOP;
  END LOOP;
  CLOSE c1;
END;
```

Параметры БД

✓ `select * from v$parameter where name like '%cursor%'`

✓ `select * from v$system_parameter where name like '%cursor%'`

Параметр	Описание
open_cursors	Максимальное количество открытых курсоров в рамках одной сессии (по умолчанию 50). Максимально возможное значение 65 535.
session_cached_cursors	Максимальное количество session cursors в кэше (по умолчанию 50). Курсоры попадают в кэш в

	случае повторного разбора (parse) одного и того же SQL (включая рекурсивный)– при этом из кэша могут удаляться курсоры, к которым давно не было обращений.
cursor_space_for_time	Параметр устаревший. Пользоваться не следует. Раньше задавал, следует ли при необходимости увеличивать размер выделяемой памяти для курсора, чтобы уменьшить cpu time.

Словари данных

- ✓ Представление V$OPEN_CURSOR возвращает список кэшированных сессией курсоров
- ✓ Чтобы получить количество открытых курсоров, используйте запрос:

```
SELECT a.value,
       s.username,
       s.sid,
       s.serial#
FROM   v$sesstat  a,
       v$statname b,
       v$session  s
WHERE  a.statistic# = b.statistic#
AND    s.sid = a.sid
AND    b.name = 'opened cursors current'
```