КУРСОРЫ

В разных СУБД спектр применения курсоров различается, но чаще всего реализуется:

* использование курсоров "только для чтения" - данные нельзя будет изменить или удалить
* указание направления чтения данных - вперёд, назад, первая или последняя строка, абсолютное или относительное положение
* разделение столбцов на редактируемые и нередактируемые
* ограничение области видимости - доступ к курсору будет только внутри запроса, который его создал или в любых запросах
* создание копии данных из таблицы с целью сохранения их в неизменном виде до обращения к курсору

**Как работать с курсорами**

Есть несколько этапов, которые можно выделить при работе с курсорами:

1. Создание. Извлечения данных на этом этапе не происходит, а только создаётся текст запроса SELECT и курсору задаются параметры
2. Открытие для получения данных. Здесь данные извлекаются из таблиц
3. Чтение данных из курсора. Можно сделать, когда курсор уже заполнен
4. Закрытие. А также, возможно, освобождение занятых им ресурсов памяти (не для всех СУБД)

Целиком процесс работы с курсором выглядит так:

* когда курсор объявлен, его можно сколько угодно раз открывать и закрывать
* когда курсор открыт, можно сколько угодно раз извлекать из него строки

**1). Создание**

SELECT \* FROM employees WHERE department\_id = 20;

  Простейшую форму курсора можно представить себе как указатель на таблицу в базе данных. Например, следующее объявление курсора связывает всю таблицу employee с курсором employee\_cur:

BEGIN;

Declare anployee\_cur CURSOR for SELECT \* FROM employee;

COMMIT;

-- для использования курсоров необходимо начать транзакцию

BEGIN;

DECLARE NewCursor CURSOR

FOR

SELECT \* FROM employees WHERE department\_id = 20;

COMMIT;

Следующий пример показывает один из способов передачи имени курсора вызывающему:

CREATE TABLE test (col text);

INSERT INTO test VALUES ('123');

CREATE FUNCTION reffunc(refcursor) RETURNS refcursor AS '

BEGIN

OPEN $1 FOR SELECT col FROM test;

RETURN $1;

END;

' LANGUAGE plpgsql;

BEGIN;

SELECT reffunc('funccursor');

FETCH ALL IN funccursor;

COMMIT;

В следующем примере используется автоматическая генерация имени курсора:

CREATE FUNCTION reffunc2() RETURNS refcursor AS '

DECLARE

ref refcursor;

BEGIN

OPEN ref FOR SELECT col FROM test;

RETURN ref;

END;

' LANGUAGE plpgsql;

-- для использования курсоров, необходимо начать транзакцию

BEGIN;

SELECT reffunc2();

FETCH ALL IN "<unnamed cursor 1>";

COMMIT;

В следующем примере показан один из способов вернуть несколько курсоров из одной функции:

CREATE OR REPLACE FUNCTION myfunc(refcursor, refcursor) RETURNS SETOF refcursor AS $$

BEGIN

OPEN $1 FOR SELECT \* FROM jobs;

RETURN NEXT $1;

OPEN $2 FOR SELECT \* FROM employees;

RETURN NEXT $2;

END;

$$ LANGUAGE plpgsql;

-- для использования курсоров необходимо начать транзакцию

BEGIN;

SELECT \* FROM myfunc('a', 'b');

FETCH ALL FROM a;

FETCH ALL FROM b;

COMMIT;

|  |  |
| --- | --- |
| Отобразите c 5-го по 10-го сотрудников из таблицы «Сотрудники».  declare  cursor empcur is  select employee\_id, first\_name  from employees;    begin  for emprec in empcur  loop  if empcur%rowcount > 4 then  dbms\_output.put\_line( emprec.first\_name);  exit when empcur%rowcount > 10;  end if;  end loop;    end;  Показать должность и имя сотрудника, который работает первый день.  declare  cursor jobscur is select job\_id, job\_title from jobs;  v\_name employees.first\_name%type;  begin  for jobrec in jobscur  loop  select first\_name into v\_name  from employees  where hire\_date = ( select min(hire\_date) from employees where job\_id = jobrec.job\_id)  and job\_id = jobrec.job\_id;    dbms\_output.put\_line( jobrec.job\_title || '-' || v\_name);  end loop;  end;  Обновление зарплаты сотрудника в зависимости от отдела и процента комиссии. Если отдел - 40, увеличьте зарплату на 10%. Если отдел - 70, то на 15%, если комиссия больше 0,3%, то 5%, в противном случае 10%.  declare  cursor empcur is  select employee\_id, department\_id, commission\_pct  from employees;    v\_hike number(2);  begin  for emprec in empcur  loop  if emprec.department\_id = 40 then  v\_hike := 10;  elsif emprec.department\_id = 70 then  v\_hike := 15;  elsif emprec.commission\_pct > 0.30 then  v\_hike := 5;  else  v\_hike := 10;  end if;  update employees set salary = salary + salary \* v\_hike/100  where employee\_id = emprec.employee\_id;    end loop;  end;  Создайте функцию, которая принимает идентификатор менеджера и возвращает имена сотрудников, которые отчитываются перед этим менеджером. Имена должны быть возвращены в виде строки с запятыми, разделяющими имена.  create or replace function get\_employees\_for\_manager(manager number)  return varchar2  is  v\_employees varchar2(1000) := '';  cursor empcur is  select first\_name from employees  where manager\_id = manager;  begin  for emprec in empcur  loop  v\_employees := v\_employees || ',' || emprec.first\_name;  end loop;  -- remove extra , at the beginning  return ltrim(v\_employees,',');  end;  ПРОВЕРЬТЕ РЕЗУЛЬТАТ РАБОТЫ ФУНКЦИИ | CREATE OR REPLACE FUNCTION k\_funk()  RETURNS SETOF employees AS  $func$  DECLARE  \_cursor CURSOR FOR SELECT \* FROM employees;  begin  OPEN \_cursor;  MOVE FORWARD 4 IN \_cursor;  RETURN QUERY FETCH 6 FROM \_cursor;  END  $func$ LANGUAGE plpgsql;  BEGIN;  SELECT k\_funk()  COMMIT; |