Выборка данных в PL/SQL

- **Неявные курсоры**
- **Явные курсоры**
- **Курсорные переменные**
- **Курсорные выражения**
- **Динамические SQL-запросы**

Термины, связанные с выборкой данных

- ***Курсор*. Рабочая область SQL-оператора. Когда приложение посылает SQL-оператор серверу Oracle, сервер открывает по меньшей мере один курсор для обработки этого оператора.**
- ***Результирующий набор строк*. Набор строк, содержащий результирующие данные, определяемые SQL-инструкцией.**
- ***Неявный курсор*. Курсор, автоматически создаваемый PL/SQL при каждом выполнении DML-инструкции (INSERT, UPDATE, DELETE) или инструкции SELECT INTO.**
- ***Явный курсор*. Инструкция SELECT, явно определенная в программе как курсор.**
- ***Курсорная переменная*. Объявленная переменная, указывающая на объект курсора. Ее значение во время выполнения программы может меняться. Может передаваться в качестве параметра процедуре или функции.**
- ***Атрибут курсора*. Внутренняя переменная Oracle, возвращающая информацию о состоянии курсора. Имеет формы *%имя_атрибута* и добавляется к имени курсора или курсорной переменной.**
- ***Инструкция SELECT…FOR UPDATE*. Разновидность инструкции SELECT, устанавливающая блокировку на каждую возвращаемую запросом строку данных. Запрещает другим пользователям изменять данные, пока вы с ними работаете.**
- ***Пакетная обработка*. Поддерживает запросы с предложением BULK COLLECT (начиная с Oracle8i). Позволяет за один раз загрузить из базы данных более одной строки в коллекции PL/SQL.**

Атрибуты курсора

Наименование	Возвращаемое значение
%FOUND	TRUE, если успешно выбрана хотя бы одна строка
%NOTFOUND	TRUE, если инструкция не выбрала ни одной строки
%ROWCOUNT	Количество строк, выбранных из курсора на данный момент
%ISOPEN	TRUE, если курсор открыт
%BULK_ROWCOUNT	Коллекция, в которой для каждого элемента исходной коллекции оператора FORALL указано количество строк, обработанных SQL-инструкцией
%BULK_EXCEPTIONS	Коллекция, в которой для каждого элемента исходной коллекции оператора FORALL, вызвавшего программную ошибку, указано инициированное исключение

Атрибуты курсоров нельзя применять в SQL-инструкциях.

1. Неявные курсоры

PL/SQL автоматически создает неявный курсор для каждой выполняемой инструкции DML (INSERT, UPDATE, DELETE) или инструкции SELECT INTO. Курсор называется неявным, т.к. Oracle автоматически выполняет многие связанные с ним операции. Для обращения к такому курсору используется ключевое слово SQL.

Неявный курсор создается для инструкции SELECT, которая

- **определяется в исполняемом разделе, а не в разделе объявлений**
- **содержит предложение INTO, которое является частью языка PL/SQL, а не SQL**
- **не требует открытия, выборки данных и закрытия.**

SELECT *список_столбцов* [BULK COLLECT] INTO *список_переменных*
. . . *оставшаяся часть инструкции SELECT*

```
DECLARE
        l_sal   sal%ROWTYPE;
BEGIN
        SELECT  *  INTO  l_sal  FROM  sal  WHERE  snum = 1001;
END;
```

Инструкция SELECT INTO (без BULK COLLECT) должна возвращать одну строку. Oracle инициирует исключение, если

- **по запросу не найдено ни одной строки – NO_DATA_FOUND**
- **возвращено несколько строк – TOO_MANY_ROWS**

```
CREATE FUNCTION sal_name (p_SNum sal.snum%TYPE)
        RETURN sal.sname%TYPE
IS
   ret_val      sal.sname%TYPE;
BEGIN
   SELECT  sname  INTO  ret_val  FROM  sal  WHERE  snum= p_SNum;
   RETURN  ret_val;
EXCEPTION
   WHEN  NO_DATA_FOUND    THEN
      RETURN  null;
   WHEN  TOO_MANY_ROWS    THEN
      Save_log ('Нарушение целостности для snum='|| p_SNum);
      RAISE;
END;
```

SQL%FOUND	**TRUE, если успешно выбрана одна строка (несколько строк для BULK COLLECT INTO)**
SQL%NOTFOUND	**TRUE, если не выбрано ни одной строки (инициируется исключение NO_DATA_FOUND)**
SQL%ROWCOUNT	**Количество выбранных строк**
SQL%ISOPEN	**Всегда FALSE**

2. Явные курсоры

Явный курсор – это инструкция SELECT, явно определенная в разделе объявлений как курсор. При объявлении курсору присваивается имя.
Для INSERT, UPDATE и DELETE явные курсоры создавать нельзя.

2.1. Объявление явного курсора

CURSOR *имя_курсора* [(*параметр* [, *параметр* …])]
[RETURN *спецификация_return*]
IS *инструкция_SELECT*
[FOR UPDATE [OF *список_столбцов*]];

- **Курсор без параметров**
  CURSOR c_ord IS
  SELECT odate FROM ord;

- **Курсор с параметрами**
  CURSOR c_sal (p_SNum NUMBER) IS
  SELECT sname FROM sal WHERE snum = p_SNum;

- **Курсор с предложением RETURN**
  CURSOR c_cust RETURN cust%ROWTYPE IS
  SELECT * FROM cust WHERE city = 'London';

Курсор не является переменной. Это идентификатор запроса. Имени курсора нельзя присвоить значение и его нельзя использовать в выражениях.

Курсор можно объявить на уровне пакета.

```
CREATE OR REPLACE PACKAGE pack_utl IS
  CURSOR c_sal IS
    SELECT sname FROM sal;
END;
```

При объявлении курсора в пакете можно отделить заголовок курсора от его тела.

```
CREATE OR REPLACE PACKAGE pack_utl IS
  CURSOR c_sal (p_SNum sal.snum%TYPE)
    RETURN sal%ROWTYPE;
END;

CREATE OR REPLACE PACKAGE BODY pack_utl IS
  CURSOR c_sal (p_SNum sal.snum%TYPE)
    RETURN sal%ROWTYPE
  IS
    SELECT * FROM sal WHERE snum = p_SNum;
END;
```

- **Сокрытие информации**
- **Минимизация перекомпиляций**

2.2. Открытие явного курсора

OPEN *имя_курсора* [(*аргумент* [, *аргумент* …])];

Оператор OPEN выполняет содержащийся в курсоре запрос. Он формирует результирующий набор строк, но не извлекает данные – это делает оператор FETCH.

```
IF NOT c_sal%ISOPEN THEN
  OPEN c_sal;
END IF;
```

2.3. Выборка данных из явного курсора

FETCH *имя_курсора* INTO *запись_или_список_переменных*;

- **Выборка данных из курсора в переменную**

```
DECLARE
   CURSOR c_ord IS
     SELECT odate FROM ord;
   ord_date   ord.odate%TYPE;
BEGIN
  OPEN c_ord;
  FETCH c_ord INTO ord_date;
```

- **Выборка данных из курсора в запись**

```
DECLARE
   CURSOR c_ord IS
     SELECT * FROM ord;
   ord_rec   ord%ROWTYPE;
BEGIN
  OPEN c_ord;
  FETCH c_ord INTO ord_rec;
```

Псевдоним столбца – это альтернативное имя в инструкции SELECT для столбца или выражения. Для явного курсора псевдонимы столбцов необходимы:

- **когда данные выбираются из курсора в запись, объявленную с атрибутом %ROWTYPE на основе того же курсора;**
- **когда в программе имеется ссылка на вычисляемый столбец.**

```
DECLARE
   CURSOR c_ord IS
     SELECT odate, SUM(amt) amt_sum FROM ord;
   ord_rec   c_ord %ROWTYPE;
BEGIN
  OPEN c_ord;
  LOOP
    FETCH c_ord INTO ord_rec;
       EXIT WHEN c_ord%NOTFOUND;
    DBMS_OUTPUT/PUT_LINE(to_char(ord_rec.odate,'dd.mm.yyyy'))||
       '  '|| ord_rec. amt_sum);
  END LOOP;
END;
```

2.4. Закрытие явного курсора

CLOSE *имя_курсора*;

- **Если курсор объявлен и открыт в процедуре, следует его закрыть после завершения работы с ним. Иначе возможна «утечка памяти»**
- **Курсор, объявленный в пакете, следует сразу же закрыть после завершения работы с ним, т.к. он остается открытым до явного закрытия или до конца сеанса**
- **Важно как можно раньше закрывать курсор с запросом SELECT FOR UPDATE, т.к. для таких запросов происходит блокировка выбираемых строк**
- **Курсор следует закрывать только в том случае, если он открыт**

```
IF c_sal%ISOPEN  THEN
   CLOSE c_sal;
END IF;
```

- **Если оставить открытыми много курсоров, можно превысить значение параметра базы данных OPEN_CURSORS. Появится сообщение об ошибке:**

```
ORA-01000:  maximum open cursors exceeded
```

2.5. Атрибуты явных курсоров

Oracle поддерживает четыре атрибута для явных курсоров: %ISOPEN, %FOUND, %NOTFOUND и %ROWCOUNT.

	%ISOPEN	%FOUND	%NOTFOUND	%ROWCOUNT
До OPEN	FALSE	Исключение	Исключение	Исключение
После OPEN и до первой FETCH	TRUE	NULL	NULL	0
После первой FETCH	TRUE	TRUE	FALSE	1
Перед последней FETCH	TRUE	TRUE	FALSE	Зависит от данных
После последней FETCH и перед CLOSE	TRUE	FALSE	TRUE	Зависит от данных
После CLOSE	FALSE	Исключение	Исключение	Исключение

2.6. Параметры курсора

- **Расширение возможности многократного использования курсоров**
- **Решение проблем, связанных с областью действия курсоров**

Область действия параметров курсора ограничивается самим курсором.
Параметры курсоров могут быть только входными.
Параметрам могут присваиваться значения по умолчанию.

```
CURSOR c_sal (p_SNum NUMBER := 0)
   RETURN  sal%ROWTYPE
IS
   SELECT  *  FROM  sal   WHERE  snum = p_SNum;
```

2.7. Предложение BULK COLLECT

Конструкция BULK COLLECT была введена в Oracle8i для ускорения работы с запросами. BULK COLLECT позволяет использовать неявный курсор для считывания за один раз нескольких строк в коллекции PL/SQL. Эта конструкция также позволяет за одно обращения к явному курсору считать в коллекции все или часть строк, выбранных курсором из базы данных.

. . . BULK COLLECT INTO *имя_коллекции*[, *имя_коллекции*] . . . [LIMIT *строки*]

```
DECLARE
   TYPE  t_t_var  IS  TABLE  OF  varchar2(20)
      INDEX BY binary_integer;
   tName    t_t_var;
   tCity      t_t_var;
BEGIN
   SELECT  sname, city  BULK COLLECT  INTO   tName, tCity
      FROM sal  WHERE  comm < 0.2;
   -- работа с коллекциями
END;
```

```
DECLARE
   CURSOR c_sal(p_Comm number)  IS
      SELECT  *  FROM sal  WHERE  comm > p_Comm;
   TYPE  t_t_sal  IS  TABLE  OF  c_sal %ROWTYPE
      INDEX BY binary_integer;
   tSal       t_t_sal;
BEGIN
   OPEN  c_sal( 0.1);
   FETCH  c_sal   BULK COLLECT  INTO   tSal   LIMIT 5;
   CLOSE  c_sal;
   -- работа с коллекцией
END;
```

- **Ключевые слова BULK COLLECT можно использовать только в предложениях SELECT INTO, FETCH INTO и RETURNING INTO**
- **Начиная с Oracle9i BULK COLLECT можно применять в динамическом SQL**
- **Коллекции строк можно применять в BULK COLLECT только начиная с Oracle9i Release2**
- **Коллекции заполняются начиная с индекса 1, и далее элементы добавляются последовательно (без пропусков), заменяя уже имеющиеся**
- **SELECT … BULK COLLECT нельзя использовать в операторе FORALL**

```
FORALL idx  IN  tName.first .. tName.last
   UPDATE  sal  SET  comm = comm + 0.05  WHERE  sname = tName(idx)
     RETURNING  comm  BULK COLLECT  INTO   tComm;
```

RETURNING в операторе FORALL заполняет коллекцию tComm последовательно для всех операторов UPDATE. RETURNING в операторе FOR заполняет коллекцию заново для каждого UPDATE, затирая уже имеющиеся элементы коллекции.

2.7. SELECT с предложением FOR UPDATE

Инструкция SELECT с предложением FOR UPDATE заблокирует набор выбранных строк еще до того, как вы приступите к их изменению. Никто не сможет изменить эти строки, пока вы не выполните COMMIT или ROLLBACK.

```
CURSOR c_sal(p_Comm number)  IS
   SELECT  *  FROM sal  WHERE  comm.> p_Comm
      FOR  UPDATE;

CURSOR c_sal(p_Comm number)  IS
   SELECT  s.sname, o.odate, o.amt
      FROM sal  s, ord o
      WHERE  s.snum=o.snum  AND  s.comm> p_Comm
      FOR  UPDATE  OF   o.amt;
```

После выполнения оператора COMMIT или ROLLBACK текущая позиция в открытом курсоре будет утеряна. Последующее выполнение оператора FETCH вызовет исключительную ситуацию.

В инструкциях UPDATE и DELETE можно использовать специальное предложение WHERE CURRENT OF для обработки последней выбранной из курсора строки.

… WHERE CURRENT OF *имя_курсора*

2.8. Курсорные переменные

Курсорная переменная – это переменная, содержащая ссылку на курсор. Курсорные переменные предоставляют механизм передачи результатов запроса между разными программами PL/SQL.

Все возможности статических курсоров и методы работы с ними применимы при использовании курсорных переменных. Используются операторы OPEN, FETCH и CLOSE. Доступны стандартные атрибуты курсоров – %ISOPEN, %FOUND, %NOTFOUND и %ROWCOUNT.

Курсорная переменная предоставляет программисту новые возможности:

- **может быть связана с разными запросами по ходу выполнения программы**
- **может быть передана в качестве аргумента процедуре или функции**
- **значение одной курсорной переменной может быть присвоено другой курсорной переменной**

2.8.1. Объявление типов REF CURSOR

TYPE *имя_типа_курсора* IS REF CURSOR [RETURN *возвращаемый_тип*];

```
TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
TYPE cust_curtype IS REF CURSOR;
```

2.8.2. Объявление курсорной переменной

имя_курсорной переменной   имя_типа_курсора **;**

```
DECLARE
    TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
    cv_sal  sal_curtype;  -- курсорная переменная
BEGIN
. . .
END;
```

2.8.3. Открытие курсорной переменной

OPEN *имя_курсорной_переменной* FOR *инструкция_select*;

Оператор OPEN инициализирует курсорную переменную, то есть создает объект курсора и помещает ссылку на него в переменную.

- **Открытие курсорной переменной сильнотипизированного типа**

```
DECLARE
    TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
    cv_sal  sal_curtype;  -- курсорная переменная
BEGIN
    OPEN  cv_sal  FOR  select  *  from  sal;
END;
```

- **Открытие курсорной переменной слаботипизированного типа**

```
DECLARE
   TYPE sal_curtype IS REF CURSOR;
   cv_sal  sal_curtype;  -- курсорная переменная
BEGIN
   OPEN  cv_sal  FOR  select  *  from  sal;
   . . .
   OPEN  cv_sal  FOR  select  sname, comm  from  sal;
   . . .
   OPEN  cv_sal  FOR  select  odate, amt  from  ord;
   . . .
END;
```

2.8.4. Выборка данных из курсорной переменной

FETCH *имя_курсорной_переменной* INTO *имя_записи;*
FETCH *имя_курсорной_переменной* INTO *имя_переменной, имя_переменной;*

Если возвращаемые запросом данные не соответствуют структурам, указанным в предложении INTO, то генерируется исключение ROWTYPE_MISMATCH.

```
DECLARE
   TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
   cv_sal   sal_curtype;  -- курсорная переменная
   rec_sal  sal%ROWTYPE;
BEGIN
   OPEN  cv_sal  FOR  select  *  from  sal;
   FETCH  cv_sal  INTO rec_sal ;
   CLOSE  cv_sal;
   . . .
END;
```

Области видимости курсорной переменной и объекта курсора могут не совпадать. Объект курсора остается доступным до тех пор, пока на него ссылается хоть одна активная переменная.

```
DECLARE
   TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
   cv_sal1   sal_curtype;  -- курсорная переменная 1
   rec_sal  sal%ROWTYPE;
BEGIN
   DECLARE
      cv_sal2   sal_curtype;  -- курсорная переменная 2
   BEGIN
      OPEN  cv_sal2  FOR  select  *  from  sal;
      cv_sal1  :=  cv_sal2;
   END;
   FETCH  cv_sal1  INTO rec_sal ;
   CLOSE  cv_sal1;
END;
```

2.8.5. Передача курсорной переменной в качестве параметра

- **Создание локального модуля внутри программы.**

```
DECLARE
    TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
    PROCEDURE open_query (p_CV  OUT  sal_curtype)  IS
        cv_sal    sal_curtype;
    BEGIN
        OPEN  cv_sal2  FOR  select  *  from  sal;
        p_CV  :=  cv_sal;
    END;
    . . .
BEGIN
    . . .
END;
```

- **Создание отдельной процедуры.**

```
PACKAGE pack_lib  IS
    TYPE sal_curtype IS REF CURSOR  RETURN sal%ROWTYPE;
END;
PROCEDURE open_query (p_CV  OUT  pack_lib.sal_curtype)  IS
    cv_sal    pack_lib.sal_curtype;
BEGIN
    . . .
END;
```

2.9. Курсорные выражения

Курсорное выражение – это выражение, включающее специальный оператор CURSOR и определяющее вложенный курсор.

- **Объявления явных курсоров**
- **Объявления и переменные типа REF CURSOR**
- **Динамические SQL-запросы**

```
DECLARE
   CURSOR c_sal  IS
      SELECT snum, sname  FROM  sal;
   CURSOR c_ord(p_Sal NUMBER)  IS
      SELECT odate,amt FROM  ord  WHERE snum = p_Sal;
BEGIN
   FOR v_sal  IN  c_sal  LOOP
      FOR v_ord  IN  c_ord(v_sal. snum)  LOOP
         DBMS_OUTPUT.PUT_LINE
            (v_sal.sname ||' '|| v_ord.odate ||' '|| v_ord.amt);
      END LOOP;
   END LOOP;
END;
```

```
DECLARE
   TYPE ord_curtype IS REF CURSOR;
   CURSOR c_sal_ord  IS
      SELECT s.sname,
                CURSOR ( SELECT o.odate,o.amt FROM  ord o
                              WHERE o.snum = s.snum)
         FROM  sal s;
   ord_cur    ord_curtype;
   v_name     sal.sname%TYPE;
   v_date     ord.odate%TYPE;
   v_sum      ord.amt%TYPE;
BEGIN
   OPEN c_sal_ord;
   LOOP
      FETCH  c_sal_ord  INTO  v_name,  ord_cur;
      EXIT  WHEN  c_sal_ord%NOTFOUND;
      LOOP
         FETCH  ord_cur  INTO  v_date ,v_sum;
         EXIT  WHEN  ord_cur %NOTFOUND;
         DBMS_OUTPUT.PUT_LINE(v_name ||' '||v_date ||' '||v_sum);
      END LOOP;
   END LOOP;
   CLOSE c_sal_ord;
END;
```