COMPASS PLUS

Обогащая традиции новыми технологиями

PL/SQL ВВЕДЕНИЕ
IT4EVA

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Procedural Language SQL

- Переменные и типы
- Управляющие структуры, такие как операторы и циклы IF-THEN-ELSE
- Процедуры и функции
- Объектные типы и методы

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Procedural Language SQL

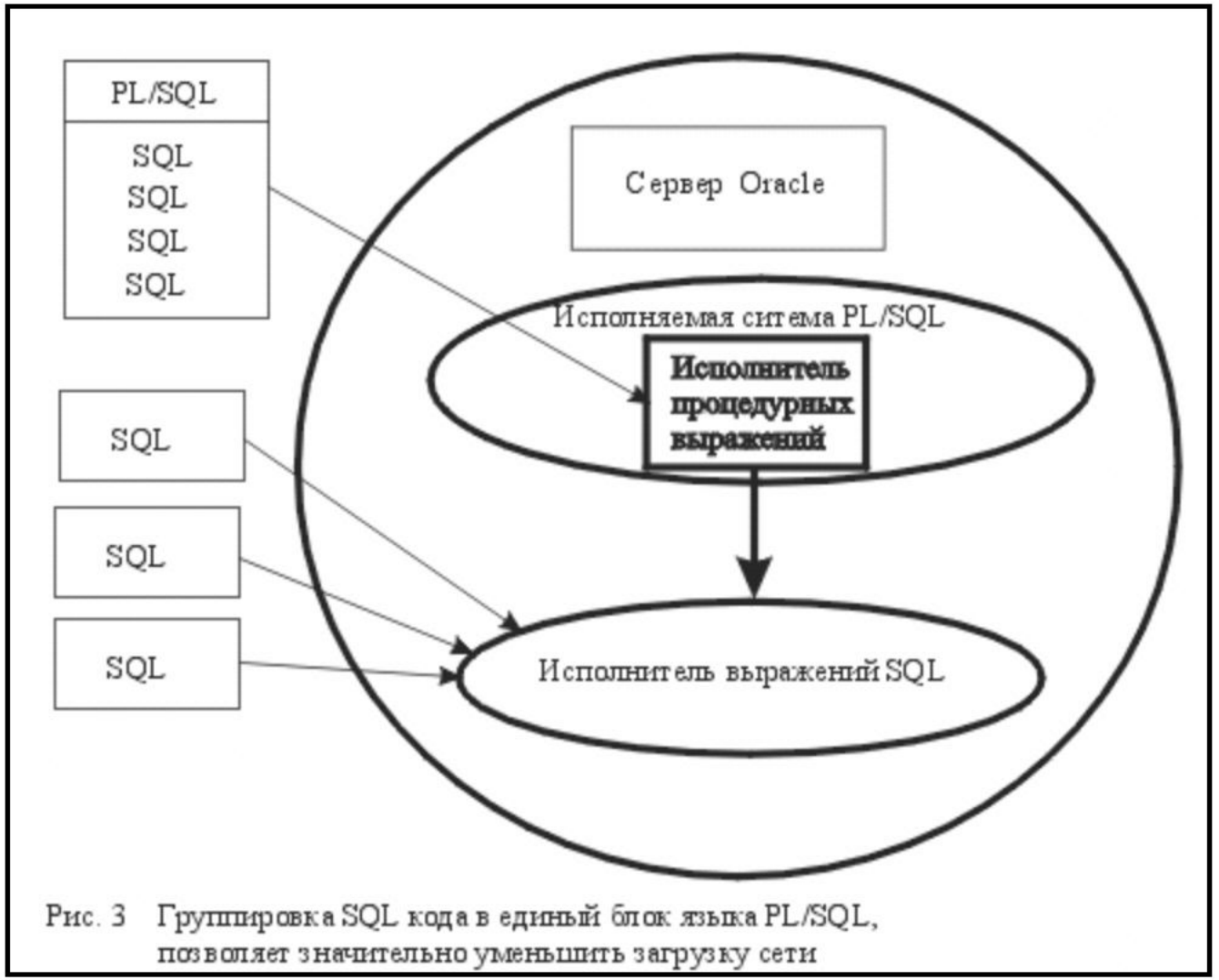

Рис. 3 Группировка SQL кода в единый блок языка PL/SQL, позволяет значительно уменьшить загрузку сети

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL:переменные, операторы, типы, блоки

- **Анонимные блоки**
  - [Declare
    …объявления]
    Begin
    [*1.. N исполняемых операторов*]
    [EXCEPTION
    … *операторы обработки исключений…*]
    end;
    / -- *символ завершения для запуска блока на компиляцию*
- **Литералы –** это значение, которое не представлено идентификатором, т. е. не имеет имени
  - **Строковый литерал –** всегда заключен в одинарные кавычки
    'This is my sentence', '01-OCT-1986', 'hello!', NULL
  - **Числовой литерал**
    415, 21.6, 3.141592654f, 7D, NULL
  - **Логический литерал**
    TRUE, FALSE, NULL

Таблица 1.1. Символы из набора US7ASCII, доступные в PL/SQL

Тип	Символы
Буквы	A–Z, a–z
Цифры	0–9
Символы	~!@#$%*()_-+=|:;»'<>,.?/^
Пробельные символы	Знак табуляции, знак пробела, перевод каретки, конец строки

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: переменные, операторы, типы, блоки

- **Операторы**
  - **Оператор** - Это элемент языка, который задает полное описание некоторого действия, которое необходимо выполнить. Каждый оператор представляет собой законченную фразу языка программирования и определяет некоторый этап обработки данных, например:

```
B := 5; X := 10;
if B = X then  DBMS_OUTPUT.put_line('Wrong result!');
end if;
```

  - Арифметические операторы: **+ - * / **** и т.д.
  - Операторы отношения (используются в логических выражениях): **< > = !=** и т.д.
  - Выражение и списки (используются в операторах, объявлениях типов данных, объявлениях списков параметров, ссылках на переменные и таблицы): **:= ( ) , ‘ ||** и т.д.
  - Комментарии и метки:

--	Комментарий в одной строке
/*	Начало многострочного комментария
*/	Конец многострочного комментария
>>	Начало метки
<<	Конец метки

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Таблица 1.2. Простые и составные специальные символы в PL/SQL

Символ	Описание
;	Точка с запятой завершает объявления и операторы.
%	Знак процента является указателем атрибутов (атрибуты курсора, такие как %ISOPEN и атрибуты косвенного объявления, как %ROWTYPE); также используется как многобайтный групповой символ в условии LIKE.
_	Одиночный символ подчеркивания: одиночный групповой символ в условии LIKE.
@	Знак @ указывает на удаленное местоположение.
:	Двоеточие является указателем хост-переменной, как :block.item в Oracle Forms.
**	Двойная звездочка – это оператор возведения в степень.
<> или != или ^= или ~=	Способы обозначения оператора отношения «не равно».
||	Двойная вертикальная черта – это знак операции конкатенации.
<< и >>	Разделители меток.
<= и >=	Операторы отношений «меньше или равно» и «больше или равно».
:=	Оператор присваивания.
=>	Оператор связывания для связывания по имени.
..	Две точки – оператор диапазона.
--	Двойной дефис служит указателем однострочного комментария.
/* и */	Начальный и конечный ограничители многострочного комментария.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Разделитель «точка с запятой»

Программа PL/SQL состоит из последовательности объявлений и операторов, границы которых определяются не физически, а логически. Другими словами, они не заканчиваются вместе с физическим концом строки кода, а завершаются символом «точка с запятой» (;). Один оператор для удобства восприятия часто записывается на нескольких строках. Например, следующий оператор IF занимает четыре строки, причем для более явного отображения логики в записи используются отступы:

```
IF salary < min_salary(2003)
THEN
   salary := salary + salary * .25;
END IF;
```

В этом операторе IF вы видите две точки с запятой. Первая точка с запятой указывает на конец отдельного исполняемого оператора внутри конструкции IF-END. Вторая точка с запятой обозначает конец самого оператора IF.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Идентификаторы**

Идентифика́тор, **ID** (англ. *data name, identifier* — опознаватель) — уникальный признак объекта, позволяющий отличать его от других объектов, то есть идентифицировать.

По умолчанию идентификаторы(переменные, типы данных, объекты, пакеты) должны обладать следующими свойствами:

- Иметь длину до 30 символов
- Должны начинаться с буквы
- Могут включать в себя знаки доллара $, подчеркивания (_) и диеза (#)
- Не могут содержать никакие пробельные символы

Например: x, v1, v1_, v_Student, vID#

Если два идентификатора отличаются только регистром одной или нескольких букв, то PL/SQL воспринимает их как один и тот же идентификатор. Например, следующие идентификаторы для PL/SQL идентичны:

lots_of_$MONEY$ LOTS_of_$MONEY$ Lots_of_$Money$

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: переменные, операторы, типы, блоки

- **Переменные**

Имя_переменной тип [CONSTANT] [NOT NULL] [:= *значение*];

vCustomerName(имя переменной) char(100) (тип данных переменной);-- признак конца оператора

vCountStudents number; sDateBirth date
sTotalAmount number(9,2); cIdentContract constant varchar2(15) := 'Licence';

- **Значение NULL**

Отсутствие значения отображается в Oracle при помощи ключевого слова NULL.
В предыдущем разделе было показано, что переменные почти всех типов данных PL/SQL могут находиться в неопределенном состоянии (исключением являются ассоциативные массивы, экземпляры которых ни при каких условиях не могут быть неопределенными). Обработка значений NULL любого типа данных может вызывать определенные сложности у программиста, при этом строковые значения заслуживают особого упоминания.
В Oracle SQL и PL/SQL строковое значение NULL обычно неотличимо от литерала, состоящего из нулевого количества символов (" – две последовательные одинарные кавычки, между которыми нет никаких символов).
Например, следующее выражение будет вычислено как TRUE и в SQL, и в PL/SQL:

" IS NULL

Значение NULL ведет себя так, как если бы типом данных по умолчанию для него являлся VARCHAR2, но сервер Oracle будет пытаться выполнить его неявное преобразование к типу данных, соответствующему выполняемой операции. В некоторых ситуациях от вас может потребоваться явное приведение типов (с использованием такой синтаксической конструкции, как TO_NUMBER(NULL) или CAST(NULL AS NUMBER)).

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Типы данных**
  - **CHAR(n), VARCHAR2(n)** - Строки переменной длины.
  - **INTEGER** - Целое число.
  - **NUMBER(n)** - Масштабируемое целое с плавающей точкой.
  - **DATE** - Дата/Время
  - **BOOLEAN** – логические выражения
  - **ROWID, ROW** - Идентификаторы записей в БД.
  - **BLOB** - Большие двоичные объекты.
  - **CLOB** - Большие строковые объекты.
  - **BFILE** - Указатели на большие внешние объекты.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Типы данных**
  - **CHAR(n), VARCHAR2(n)** - Строки переменной длины.

CHAR и NCHAR – это типы строк фиксированной длины,

а VARCHAR2 и NVARCHAR2 – типы строк переменной длины.

Рассмотрим объявление строки переменной длины, которая может вмещать до 2000 символов:

```
DECLARE
  l_accident_description VARCHAR2(2000);
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Типы данных**
  - **INTEGER** – целое число - целые числа в диапазоне от -2,147,483,648 до 2,147,483,647 .

vIndex integer;

  - **NUMBER(n)** - целое с плавающей точкой.

```
DECLARE
   salary NUMBER(9,2); -- фиксированная точка, семь знаков слева и два справа
   raise_factor NUMBER; -- десятичное число с плавающей точкой
   weeks_to_pay NUMBER(2); -- целое число
BEGIN
   salary := 1234567.89;
   raise_factor := 0.05;
   weeks_to_pay := 52;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Типы данных**

  - **DATE** - Дата/Время

vBirthDate date;
date может принимать значения от 1 января 4712 года до н.э. до 31 декабря 9999 года нашей эры.

  - **BOOLEAN** – логические выражения

vFirstRow boolean:= false;
Переменная этого типа может иметь лишь одно из трех значений: TRUE, FALSE и NULL.

```
Declare
  vAmount1 number := 10;
  vAmount2 number := 1;
  vResult     boolean;
Begin
  if vAmount1= vAmount2 then
    DBMS_OUTPUT.PUT_LINE('FALSE!');
    vResult := false;
  Else
    vResult := true;
  end if;
end;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Типы данных**
  - **%ROWTYPE** – курсорная переменная

```
sRowEMP EMP%RowType; - строка из таблицы EMP
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Составные типы данных**
  - **Таблицы**

TYPE TNumTable is Table of number Index by BINARY_INTEGER;

  - **Записи**

TYPE tRecStudent is record (Name varchar2(240), StudentID number);

TYPE TTabStudents is Table of tRecStudent Index by BINARY_INTEGER;

vaStudents TTabStudents;

vaStudents(1).Name – значение столбца Name 1 строки таблицы

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: условные операторы

- **Оператор IF-THEN-ELSE**

```
if <некоторое условие справедливо> then
<действия над переменными>
else – условие не выполнено
<действия над переменными>
end if;
```

```
if <некоторое условие справедливо> then
<действия над переменными>
Elsif <следующее условие справедливо> then
<действия над переменными>
else – условие не выполнено
<действия над переменными>
end if;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- Оператор IF-THEN-ELSE

```
IF( a AND b ) THEN      -- проверка условия
        BEGIN
                .
                .
        END;
END IF; -- конец условного оператора.
```

```
IF( (a OR f) AND (c OR k) ) THEN        -- проверка условия
        BEGIN
                .
                .
        END;
END IF; -- конец условного оператора.
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Простой оператор CASE**

```
CASE employee_type
  WHEN 'S' THEN          award_salary_bonus(employee_id);
  WHEN 'H' THEN          award_hourly_bonus(employee_id);
  WHEN 'C' THEN          award_commissioned_bonus(employee_id);
ELSE
  RAISE invalid_employee_type;
END CASE;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- Поисковый оператор CASE

```
CASE
WHEN salary >= 10000 AND salary <=20000 THEN
   give_bonus(employee_id, 1500);
WHEN salary > 20000 AND salary <= 40000 THEN
   give_bonus(employee_id, 1000);
WHEN salary > 40000 THEN
   give_bonus(employee_id, 500);
ELSE
   give_bonus(employee_id, 0);
END CASE;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- Простой цикл LOOP

```
PROCEDURE display_multiple_years (
   start_year_in IN PLS_INTEGER
  ,end_year_in IN PLS_INTEGER
)
IS
   l_current_year PLS_INTEGER := start_year_in;
BEGIN
   LOOP
      EXIT WHEN l_current_year > end_year_in;
      display_total_sales (l_current_year);
      l_current_year :=  l_current_year + 1;
   END LOOP;
END display_multiple_years;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- цикл FOR

```
For k in 1..vCount loop
  <действия с переменными>
end loop;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- цикл FOR

```
For k in (select * from EMP where job = ‘MANAGER’) loop
  <действия с переменными>
end loop;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

– цикл WHILE

```
PROCEDURE display_multiple_years (
   start_year_in IN PLS_INTEGER
  ,end_year_in IN PLS_INTEGER
)
IS
   l_current_year PLS_INTEGER := start_year_in;
BEGIN
   WHILE (l_current_year <= end_year_in)
   LOOP
      display_total_sales (l_current_year);
      l_current_year :=  l_current_year + 1;
   END LOOP;
END display_multiple_years;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- SELECT name FROM emp;

- CURSOR curs_get_emp
  IS
  SELECT name FROM emp;

Курсор – это средство извлечения данных из базы данных Oracle. *Курсоры* содержат определения столбцов и объектов (таблиц, представлений и т. п.), из которых будут извлекаться данные, а также набор критериев, определяющих, какие именно строки должны быть выбраны.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
Cursor cEmployGrade
 is
select  e.empno  emp_no ,
        e.ename  emp_name,
        e.sal    emp_sal,
        g.grade  emp_grade
 from emp e
 inner join salgrade g on e.sal between g.losal and g.hisal
where e.deptno = p_dept_no
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
CURSOR имя_курсора [ ( [ параметр [, параметр ...] ) ]
   [ RETURN спецификация_возврата ]
   IS SELECT_оператор
       [FOR UPDATE [OF [список_столбцов]];
```

где *имя_курсора* – имя курсора, *спецификация_возврата* – необязательное предложение RETURN для курсора, а *SELECT_оператор* – любой корректный SQL-оператор SELECT. Вы также можете передавать в курсор параметры, используя необязательный список параметров. Как только курсор объявлен, вы можете открывать его и выбирать из него данные.

Oracle поддерживает для явных курсоров тот же набор атрибутов, что и для неявных курсоров. Значения, которые атрибуты явных курсоров могут приобретать до и после выполнения указанных операций над курсорами, приведены в табл. 1.7.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Полный и частичный разбор

Процесс компиляции нового курсора называется *полным разбором* (и сам по себе заслуживает отдельной книги); в контексте этой главы он может быть упрощенно представлен четырьмя этапами:

Проверка

Курсор проверяется на соответствие синтаксическим правилам SQL, также проверяются объекты (таблицы и столбцы), на которые он ссылается.

Компиляция

Курсор компилируется в исполняемый вид и загружается в разделяемый пул сервера базы данных. Для определения местоположения курсора в разделяемом пуле используется его *адрес*.

Вычисление плана выполнения

Оптимизатор по стоимости (cost-based optimizer – CBO) Oracle определяет наилучший для данного курсора план выполнения и присоединяет его к курсору.

Вычисление хеша

ASCII-значения всех символов курсора складываются и передаются в функцию хеширования. Эта функция рассчитывает значение, по которому курсор легко может быть найден при повторном обращении. Данное значение называется *хеш-значением* курсора. Ниже в этом разделе мы еще вернемся к ASCII-значениям.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Как Oracle принимает решение о совместном использовании

Чтобы определить, может ли планируемый к выполнению курсор воспользоваться уже скомпилированной версией из разделяемого пула, Oracle применяет сложный алгоритм. Вот его упрощенное изложение:

1. Рассчитать сумму ASCII-значений всех символов курсора (исключая переменные связывания). Например, сумма ASCII-значений следующего курсора равна 2556:

```
SELECT order_date FROM orders
```

   Это значение рассчитывается как ASCII(S) + ASCII(E) + ASCII(L)... или 83 + 69 + 76 и т. д.
2. Применить алгоритм хеширования к полученной сумме.
3. Проверить наличие в разделяемом пуле курсора с таким же значением хеша.
4. Если такой курсор найден, он может быть использован повторно.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
SQL> SELECT order_date
  2    FROM orders
  3   WHERE order_number = 11;

ORDER_DAT
---------
03-MAY-05

SQL> SELECT order_date
  2    FROM orders
  3   WHERE order_numbeR = 11;

ORDER_DAT
---------
03-MAY-05
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
SQL> SELECT sql_text,
  2         parse_calls,
  3         executions
  4    FROM v$sql
  5   WHERE INSTR(UPPER(sql_text),'ORDERS')  > 0
  6     AND INSTR(UPPER(sql_text),'SQL_TEXT') = 0
  7     AND command_type = 3;

SQL_TEXT                       PARSE_CALLS EXECUTIONS
------------------------------ ----------- ----------
SELECT order_date   FROM order           1          1
s  WHERE order_numbeR = 11

SELECT order_date   FROM order           1          1
s  WHERE order_number = 11
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: курсоры

```
DECLARE
  CURSOR one IS
  SELECT order_date
    FROM orders
   WHERE order_number = 11;
  CURSOR two IS
  SELECT order_date
    FROM orders
   WHERE order_numbeR = 11;
  v_date DATE;
BEGIN
  -- открытие и закрытие корректного курсора
  OPEN one;
  FETCH one INTO v_date;
  CLOSE one;
  -- открытие и закрытие отличающегося курсора
  OPEN two;
  FETCH two INTO v_date;
  CLOSE two;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: курсоры

```
CREATE OR REPLACE PROCEDURE simple_demo AS
  CURSOR one IS
  SELECT order_date
    FROM orders
   WHERE order_number = 11;
  CURSOR two IS
  SELECT order_date
     FROM orders
   WHERE order_numbeR = 11;
  v_date DATE;
BEGIN
  -- открытие и закрытие корректного курсора
  OPEN one;
  FETCH one INTO v_date;
  CLOSE one;
  OPEN two;
  FETCH two INTO v_date;
  CLOSE two;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: курсоры

```
CREATE OR REPLACE PROCEDURE two_queries
AS
  v_order_date DATE;
BEGIN
  -- get order 100
  SELECT order_date
    INTO v_order_date
    FROM orders
   WHERE order_number = 100;
  -- get order 200
  SELECT order_date
    INTO v_order_date
    FROM orders
   WHERE order_number = 200;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
CREATE OR REPLACE PROCEDURE two_queries AS
  -- определяем параметризованный курсор
  CURSOR get_date ( cp_order NUMBER ) IS
  SELECT order_date
    FROM orders
   WHERE order_number = cp_order;
  v_order_date DATE;
BEGIN
  -- get order 100
  OPEN get_date(100);
  FETCH get_date INTO v_order_date;
  CLOSE get_date;
  -- get order 200
  OPEN get_date(200);
  FETCH get_date INTO v_order_date;
  CLOSE get_date;
END;
```

В разделяемом пуле после его очистки и выполнения новой функции будет следующее:

```
SQL_TEXT                          PARSE_CALLS EXECUTIONS
--------------------------------- ----------- ----------
SELECT ORDER_DATE FROM ORDERS               1          2
WHERE ORDER_NUMBER = :B1
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Синтаксический анализ

Первым этапом обработки оператора SQL является его синтаксический анализ, который проводится для проверки корректности оператора и определения плана его выполнения.

Связывание

Связывание – это сопоставление значений из вашей программы (хост-переменных) заполнителям используемого оператора SQL.

Для статического SQL такое связывание выполняет само ядро PL/SQL. Для динамического SQL программист, если он планирует использовать переменные связывания, должен явно запросить выполнение этой операции.

Открытие

При открытии курсора переменные связывания используются для определения результирующего множества команды SQL. Указатель активной (текущей) строки устанавливается на первой строке. В некоторых случаях явное открытие курсора не требуется; ядро PL/SQL само выполняет эту операцию (например, для неявных курсоров или встроенного динамического SQL).

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Исполнение

На этапе исполнения оператор выполняется внутри ядра SQL.

Выборка

При выполнении запроса команда FETCH извлекает следующую строку из результирующего множества курсора. При каждой выборке PL/SQL передвигает курсор вперед на одну строку по результирующему множеству. При работе с явными курсорами следует помнить, что в случае, когда строк для извлечения больше нет, FETCH ничего не делает (не инициирует исключение).

Закрытие

На этом этапе курсор закрывается, освобождается используемая им память. После закрытия курсор уже не содержит результирующее множество. В некоторых случаях явное закрытие курсора не требуется, ядро PL/SQL само выполняет эту операцию (например, для неявных курсоров или встроенного динамического SQL).

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Атрибут	Описание
%BULK_ROWCOUNT	Количество записей, возвращаемых операцией массовой выборки (BULK COLLECT INTO).
%FOUND	TRUE, если последняя операция FETCH была успешной; FALSE – в противном случае.
%NOTFOUND	TRUE, если последняя операция FETCH была неуспешной; FALSE – в противном случае.
%ISOPEN	TRUE, если курсор открыт; FALSE – в противном случае.
%ROWCOUNT	Количество записей, выбранных на текущий момент курсором.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: курсоры

```
CREATE OR REPLACE PROCEDURE demo AS
  CURSOR curs_get_date IS
  SELECT order_date
    FROM orders
   WHERE order_number = 1;
  v_date DATE;
BEGIN
  OPEN curs_get_date;
  FETCH curs_get_date INTO v_date;
  IF curs_get_date%NOTFOUND THEN
    do_something;
  END IF;
  CLOSE curs_get_date;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

язык манипулирования данными - **DML** (**Data manipulation language**) и языком определения данных - **DDL** (**Data definition language**).

- CREATE TABLE t_tbl(fld1 VARCHAR2(128));

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Управление транзакциями в PL/SQL

Как и следовало ожидать, реляционная база данных Oracle поддерживает очень мощный и надежный механизм транзакций. Код вашего приложения определяет, из чего будет состоять *транзакция* – логическая единица работы, результат которой сохраняется при помощи оператора COMMIT или отменяется оператором ROLLBACK. Транзакция неявно начинается с первого оператора SQL, выполненного после последнего оператора COMMIT или ROLLBACK (или с начала сеанса), или продолжается после ROLLBACK TO SAVEPOINT.

PL/SQL содержит ряд операторов для управления транзакциями:

COMMIT

Сохраняет все изменения, сделанные после последней операции COMMIT или ROLLBACK, и освобождает все блокировки.

ROLLBACK

Отменяет все изменения, сделанные после последней операции COMMIT или ROLLBACK, и освобождает все блокировки.

ROLLBACK TO SAVEPOINT

Отменяет все изменения, сделанные после установки указанной точки сохранения, и освобождает блокировки, которые были установлены в данном фрагменте кода.

SAVEPOINT

Устанавливает точку сохранения, которая затем позволит выполнять частичный откат.

SET TRANSACTION

Позволяет начать сеанс[1] в режиме «только для чтения» или «для чтения и записи», задать уровень изоляции или сопоставить текущей транзакции определенный сегмент отката.

LOCK TABLE

Позволяет заблокировать всю таблицу базы данных в определенном режиме. Позволяет изменить обычно применяемую к таблице установку по умолчанию – блокировку на уровне строк.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Процедуры, функции и пакеты

PL/SQL поддерживает следующие структуры, позволяющие разбить ваш код на модули:

Процедура

Программа, которая выполняет одно или несколько действий и вызывается как исполняемый оператор PL/SQL. Используя список параметров, вы можете передавать информацию в процедуру и из нее.

Функция

Программа, которая возвращает единственное значение и используется как выражение PL/SQL. Используя список параметров, вы можете передавать информацию в функцию.

Пакет

Именованная коллекция процедур, функций, типов и переменных. На самом деле пакет является скорее не модулем, а метамодулем, но без его упоминания описание модульной структуры было бы неполным.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
PROCEDURE [схема.]имя [( параметр [, параметр ...] ) ]
   [AUTHID DEFINER | CURRENT_USER]
IS
   [операторы объявления]
BEGIN
   исполняемые операторы
[ EXCEPTION
     операторы обработки исключений]
END [имя];
```

где каждый элемент имеет соответствующее назначение:

схема

Имя схемы, которой принадлежит процедура (необязательный параметр). По умолчанию процедура принадлежит схеме текущего пользователя. Для создания процедуры в другой схеме текущему пользователю потребуются соответствующие привилегии.

имя

Имя процедуры, которое указывается сразу после ключевого слова PROCEDURE**.**

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

параметры

Необязательный список параметров, которые могут быть определены для передачи информации как в процедуру, так и из нее обратно в вызывающую программу.

AUTHID ***предложение***

Определяет, с какими правами будет исполняться процедура: с правами ее владельца (создателя) или же с правами вызывающего пользователя. Принято говорить о двух моделях исполнения: *с правами владельца* и *с правами вызывающего*.

операторы объявления

Объявления локальных идентификаторов для данной процедуры. Если вы ничего не объявляете, то операторы IS и BEGIN будут следовать непосредственно друг за другом.

исполняемые операторы

Операторы, которые процедура исполняет при вызове. После ключевого слова BEGIN до ключевых слов END или EXCEPTION должен быть указан хотя бы один исполняемый оператор.

операторы обработки исключений

Необязательные обработчики исключений для процедуры. Если вы не обрабатываете явно никакие исключения, то пропустите ключевое слово EXCEPTION и завершите раздел исполнения ключевым словом END.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Вызов процедуры

Процедура вызывается как исполняемый оператор PL/SQL. Другими словами, вызов процедуры должен завершаться точкой с запятой (;) и исполняться до или после других (если они есть) операторов SQL или PL/SQL в исполняемом разделе PL/SQL-блока.

Для запуска процедуры `apply_discount` используются следующие операторы:

```
BEGIN
   apply_discount( new_company_id, 0.15 );   -- скидка 15%
END;
```

Если процедура не принимает параметров, можно вызывать ее, не используя скобки:

```
display_store_summary;
```

В версиях Oracle8*i* Database и старше можно включать в вызов процедуры пустые скобки, например:

```
display_store_summary( );
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Функции

Функция – это модуль, возвращающий значение. В отличие от вызова процедуры, являющегося независимым исполняемым оператором, вызов функции может существовать только как часть исполняемого оператора (то есть он может быть, например, элементом выражения или значением, присваиваемым по умолчанию при объявлении переменной).

Функция возвращает значение, которое, естественно, относится к какому-то типу данных. Функция может использоваться в PL/SQL-операторе вместо выражения, имеющего тот же тип данных, что и возвращаемое функцией значение.

Функции чрезвычайно важны при создании модульных конструкций. Например, любое бизнес-правило или формула в вашем приложении должны быть помещены в функции. Любой запрос, возвращающий единственную строку, также следует определять в функции, с тем чтобы обеспечить простой и надежный способ его повторного использования.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Структура функции

Структура функции совпадает со структурой процедуры, единственное отличие состоит в том, что функция включает в себя еще предложение RETURN**:**

```
FUNCTION [схема.]имя [( параметр [, параметр ...] ) ]
   RETURN тип_возвращаемых_данных
   [AUTHID DEFINER | CURRENT_USER]
   [DETERMINISTIC]
   [PARALLEL ENABLE ...]
   [PIPELINED]
IS
   [операторы объявления]

BEGIN
   исполняемые операторы

[EXCEPTION
   операторы обработки исключений]

END [ имя ];
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

DETERMINISTIC *предложение*

Подсказка оптимизатору, позволяющая системе использовать сохраненную копию возвращенного функцией результата (при его наличии). Оптимизатор запроса определяет, следует ли выбрать сохраненную копию или же вызвать функцию повторно.

PARALLEL_ENABLE *предложение*

Подсказка оптимизатору, разрешающая параллельное выполнение функции при вызове из оператора SELECT.

PIPELINED *предложение*

Указывает, что результаты табличной функции должны возвращаться построчно с помощью команды PIPE ROW.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
DECLARE
   v_nickname VARCHAR2(100) :=
      favorite_nickname('Steven');
```

- **Использование функции-члена для объектного типа** `pet` **в условном выражении:**

```
DECLARE
   my_parrot pet_t :=
      pet_t (1001, 'Mercury', 'African Grey',
             TO_DATE ('09/23/1996', 'MM/DD/YYYY'));
BEGIN
   IF my_parrot.age < INTERVAL '50' YEAR -- тип INTERVAL в 9i
   THEN
      DBMS_OUTPUT.PUT_LINE ('Still a youngster!');
   END IF;
```

- **Извлечение одной строки информации о книге непосредственно в запись:**

```
DECLARE
   my_first_book books%ROWTYPE;
BEGIN
   my_first_book := book_info.onerow ('1-56592-335-9');
   ...
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Режимы использования параметров

Определяя параметры, вы должны указать, каким образом их можно использовать. Существуют три различных режима использования параметров.

Режим	Описание	Использование параметра
IN	Только для чтения	Значение параметра может использоваться внутри модуля, но изменение параметра запрещено.
OUT	Только для записи	Модуль может присвоить параметру значение, но ссылаться на него в модуле запрещено.
IN OUT	Для чтения и записи	Модуль может ссылаться на параметр (читать) и изменять его значение (записывать).

Режим использования определяет, каким образом программа может применять значение, присвоенное формальному параметру. Режим использования параметра задается непосредственно после указания имени параметра перед указанием его типа и значения по умолчанию

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Пакеты

Пакет – это сгруппированные вместе элементы PL/SQL-кода. Пакеты представляют собой физическую и логическую структуру для организации программ и других элементов PL/SQL, таких как курсоры, типы и переменные. Они также предоставляют важные функциональные средства, такие как сокрытие логики и данных, определение глобальных данных (существующих на протяжении сеанса) и работу с ними.

```
create or replace
package emp_maint is
  procedure hire_emp(p_empno number, p_name varchar2);
  procedure fire_emp(p_empno);
  procedure raise_salary(p_empno number, p_salary number);
end;
/
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- **Спецификация пакета**
  - Вы можете объявлять элементы практически всех типов данных: числа, исключения, типы и коллекции, на уровне пакета (то есть не внутри конкретной процедуры или функции пакета). Такие данные называются данными уровня пакета. При этом следует избегать объявления переменных в пакете, тогда как объявление констант вполне допустимо.
  - Вы можете объявлять практически любые виды структур данных, такие как коллекции, записи или тип REF CURSOR
  - Вы можете объявлять в спецификации пакета процедуры и функции, но указывать можно только заголовок программы (все, что находится выше ключевого слова IS или AS).
  - Если в спецификации пакета объявлены какие-то процедуры или функции, а также если курсорная переменная объявлена без SQL запроса, то необходимо написать тело пакета, в котором будут реализованы эти элементы кода.
  - В спецификацию пакета можно включить предложение AUTHID, которое будет определять, в соответствии с какими привилегиями разрешены любые ссылки на объекты: владельца пакета (AUTHID DEFINER) или пользователя, вызывающего пакет (AUTHID CURRENT_USER).
  - После оператора END в спецификации пакета можно поместить необязательную метку с именем пакета, например: **END my_package;**

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Тело пакета

Тело пакета содержит весь код, который необходим для реализации спецификации пакета. Тело пакета требуется не всегда, но оно обязано присутствовать при наличии хотя бы одного из перечисленных далее условий:

Спецификация пакета содержит объявление курсора с предложением RETURN

Необходимо определить в теле пакета оператор SELECT.

Спецификация пакета содержит объявление процедуры или функции

Необходимо представить полную реализацию модуля в теле пакета.

Вам требуется исполнение кода в разделе инициализации тела пакета

Спецификация пакета не включает в себя исполняемый раздел (исполняемые операторы внутри конструкции BEGIN...END); вы можете исполнять код только в теле пакета.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- Тело пакета

- Тело пакета может включать в себя раздел объявлений, исполняе мый раздел и раздел исключений. Раздел объявлений содержит полную реализацию всех курсоров и программ, определенных в спе цификации, а также определение любых приватных элементов (не приведенных в спецификации). При наличии раздела инициализа ции раздел объявлений может быть пуст.
- Исполняемый раздел пакета принято называть разделом инициали зации. Он может содержать код, который исполняется при созда нии в сеансе экземпляра пакета.1
- Раздел исключений обрабатывает любые исключения, порожден ные в разделе инициализации. Он может присутствовать в конце пакета только в случае наличия раздела инициализации
- Тело пакета может быть составлено следующими способами: оно может включать в себя только раздел объявлений, только испол няемый раздел, исполняемый раздел и раздел исключений, а также раздел объявлений, исполняемый раздел и раздел исключений.
- Не разрешается использовать предложение AUTHID в теле пакета, оно должно содержаться в спецификации. В теле пакета могут использо ваться любые объекты, объявленные в спецификации этого пакета
- Все правила и ограничения, существующие для объявления струк тур данных уровня пакета, относятся как к спецификации, так и к телу пакета. Например, запрещено объявлять курсорные переменные.
- После оператора END в спецификации пакета можно поместить не обязательную метку с именем пакета, например: **END my_package;**

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

```
create or replace
package body emp_maint is

procedure hire_emp(p_empno number, p_name varchar2) is
begin
  insert into emp …
end;

procedure fire_emp(p_empno) is
begin
  delete from emp …
end;

procedure raise_salary(p_empno number, p_salary number) is
begin
  update emp …
end;
end;
/
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Данные пакета

Данные пакета представляют собой переменные и константы, которые определены *на уровне пакета* (то есть не внутри какой-то функции или процедуры пакета). Областью действия данных пакета является не какая-то одна программа, а весь пакет. Структуры данных пакета существуют (*сохраняют свои значения*) на протяжении всего сеанса, а не только во время исполнения одной программы.

Если данные пакета определены в теле пакета, то они сохраняют свои значения на протяжении сеанса, но доступ к ним разрешен только для элементов этого пакета (приватные данные).

Если данные пакета определены в спецификации пакета, то они сохраняют свои значения на протяжении сеанса и напрямую доступны (как на чтение, так и на запись) любой программе, обладающей привилегией EXECUTE для данного пакета (общие данные).

Если процедура пакета открывает курсор, то он остается открытым и доступным на протяжении всего сеанса. Нет необходимости в определении курсора в каждой программе. Один модуль может открывать курсор, а другой – выполнять выборку. Кроме того, переменные пакета могут передавать данные из одной транзакции в другую, так как переменные привязаны к сеансу, а не к какой-то определенной транзакции.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: обработка исключений

В языке PL/SQL ошибки любого рода трактуются как *исключения* – нештатные ситуации для вашей программы. Исключения могут быть следующих видов:

- Ошибка, инициированная системой (например, «недостаточно памяти» или «повторение значений в индексе»).
- Ошибка, вызванная действиями пользователя.
- Предупреждение, выдаваемое пользователю приложением.

PL/SQL перехватывает ошибки и реагирует на них, используя механизм обработчиков исключений. Обработчики исключений позволяют аккуратно отделить код обработки ошибок от исполняемых операторов. Для обработки ошибок используется событийная модель исполнения кода, а не линейная. Другими словами, вне зависимости от того, где было инициировано исключение, оно будет обработано одним и тем же обработчиком исключений в разделе исключений.

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN . . .;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

- Встроенные функции обработки ошибок

- **SQLCODE** - Функция SQLCODE возвращает код ошибки для последнего (текущего) исключения в блоке. При отсутствии ошибок SQLCODE возвращает 0. Функция SQLCODE также возвращает 0 в случае, если она вызывается извне обработчика исключений.
- **SQLERRM** - Функция SQLERRM возвращает сообщение об ошибке по ее коду. Если не передать SQLERRM код ошибки, то будет выдано сообщение об ошибке с кодом, возвращенным функцией SQLCODE. Максимальная длина строки, возвращаемой SQLERRM, составляет 512 байт (в некоторых более ранних версиях Oracle – всего 255 байт)
- **DBMS_UTILITY.FORMAT_ERROR_STACK** - Эта встроенная функция, как и SQLERRM, возвращает полное сообщение, соответствующее текущей ошибке (то есть значению, возвращенному функцией SQLCODE). Как правило, следует вызывать эту функцию внутри обработчика ошибок для того, чтобы получить полное сообщение об ошибке.
- **DBMS_UTILITY.FORMAT_ERROR_BACKTRACE** - Эта функция, появившаяся в версии Oracle Database 10g Release 1, возвращает форматированную строку, которая отображает стек программ и номера строк, ведущие к месту возникновения ошибки.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Передача необработанного исключения

Правила для области действия исключений определяют блок, в котором исключение может быть инициировано. Правила передачи исключений определяют способ их обработки после инициирования.

Когда порождается исключение, PL/SQL ищет обработчик исключений в текущем блоке (анонимном блоке, процедуре или функции). Если обработчик не найден, то PL/SQL передает исключение в родительский блок текущего блока. Затем PL/SQL пытается обработать исключение, инициировав его еще раз в родительском блоке. Процесс продолжается до тех пор, пока не закончатся все последовательные родительские блоки, в которых можно было бы инициировать исключение (рис. 1.3).

Когда все блоки исчерпаны, PL/SQL возвращает необработанное исключение в среду приложения, которое исполняло самый внешний блок PL/SQL. Необработанное исключение прекращает исполнение вызывающей программы.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: обработка исключений

```
PROCEDURE list_my_faults IS
BEGIN
   ...
   DECLARE                                                    Вложенный блок 1
      too_many_faults EXCEPTION;
   BEGIN
      ... executable statements before new block ...
      BEGIN                                                   Вложенный блок 2
        SELECT SUM (faults) INTO num_faults FROM profile ... ;
         IF num_faults > 100
         THEN
            RAISE too_many_faults;
         END IF;
      EXCEPTION
         WHEN NO_DATA_FOUND THEN ... ;
      END;
      ... executable statements after Nested Block 2 ...

   EXCEPTION
      WHEN too_many_faults THEN ... ;
   END;
END list_my_faults;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: обработка исключений

```
WHEN имя_исключения [ OR имя_исключения ... ]
THEN
   исполняемые операторы
```

или так:

```
WHEN OTHERS
THEN
   исполняемые операторы
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: обработка исключений

```
function  GetText
  (
    pDocNo in number
  )
  return varchar is
    vATMExtRow tATMExt%ROWTYPE;
  begin
    begin
      select *
        into vATMExtRow
        from tATMExt
        where Branch = sBranch and
              DocNo  = pDocNo;
    exception
      when NO_DATA_FOUND then
        select *
          into vATMExtRow
          from tATMExtArc
          where Branch = sBranch and
                DocNo  = pDocNo;
    end;
    return rpad('PAN: ' || Card.GetMaskedPAN(vATMExtRow.PAN) || '-' || to_char(vATMExtRow.
           rpad('Term(Location): ' || vATMExtRow.Term || '(' || vATMExtRow.TermLocation ||
           rpad('Amount(Currency): ' || to_char(vATMExtRow.Value) || '(' || to_char(vATMEx
  exception
    when OTHERS then
      return null;
  end;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Динамический SQL и динамический PL/SQL

Динамический SQL подразумевает под собой те операторы SQL, которые формируются и исполняются во время исполнения программы. Термин «динамический» – антоним для термина «статический». *Статический SQL* – это операторы SQL, которые фиксируются в момент компиляции программы и далее не изменяются. *Динамический PL/SQL* соответственно понимается как целые PL/SQL-блоки кода, которые динамически формируются, затем компилируются и исполняются.

Используйте оператор `EXECUTE IMMEDIATE` **для (немедленного!) исполнения указанной команды SQL:**

```
EXECUTE IMMEDIATE строка_SQL
   [INTO {переменная[, переменная]... | запись}]
   [USING [IN | OUT | IN OUT] аргумент_связывания
       [, [IN | OUT | IN OUT] аргумент_связывания]...];
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

строка_SQL

Строковое выражение, содержащее команду SQL или PL/SQL-блок.

переменная

Переменная, которая получает значение столбца, возвращенное запросом.

запись

Запись, основанная на определенном пользователем типе или атрибуте %ROWTYPE, которая получает целую строку, возвращенную запросом.

аргумент_связывания

Выражение, значение которого передается в команду SQL или PL/SQL-блок, или идентификатор, который служит входной и/или выходной переменной для функции или процедуры, вызываемой в PL/SQL-блоке.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

INTO *предложение*

Используется для однострочных запросов. Для каждого значения столбца, возвращенного запросом, следует указать отдельную переменную или поле записи совместимого типа.

USING *предложение*

Используется для передачи аргументов связывания в строку SQL. Это предложение используется как для динамического SQL, так и для динамического PL/SQL, поэтому существует возможность указать режим передачи параметров (имеет смысл только для PL/SQL; по умолчанию установлен в значение «IN», соответствующее единственному виду аргументов, доступных для команд SQL).

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: динамический SQL

Оператор EXECUTE IMMEDIATE **может использоваться для любой SQL-команды или PL/SQL-блока за исключением многострочных запросов. Если** *строка_SQL* **заканчивается точкой с запятой, то она воспринимается как PL/SQL-блок. В противном случае она воспринимается как** SELECT**, оператор DML (**INSERT**,** UPDATE **или** DELETE**) или DDL (например,** CREATE TABLE**). В строке могут присутствовать заполнители для аргументов связывания, но с их помощью нельзя передавать имена объектов схемы, такие как имена таблиц и столбцов.**

При выполнении команды исполняющее ядро заменяет каждый заполнитель (идентификатор, перед которым стоит двоеточие, например :salary_value**) в** *строке_SQL* **соответствующим** *аргументом_связывания* **(в соответствии с позицией). Вы можете передавать числа, даты и строки.**

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: динамический SQL

```
CREATE OR REPLACE FUNCTION updNVal (
   col IN VARCHAR2,
   val IN NUMBER,
   start_in IN DATE,
   end_in IN DATE)
   RETURN PLS_INTEGER
IS
BEGIN
   EXECUTE IMMEDIATE
      'UPDATE employee SET ' || col || ' = :the_value
        WHERE hire_date BETWEEN :lo AND :hi'
     USING val, start_in, end_in;
   RETURN SQL%ROWCOUNT;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

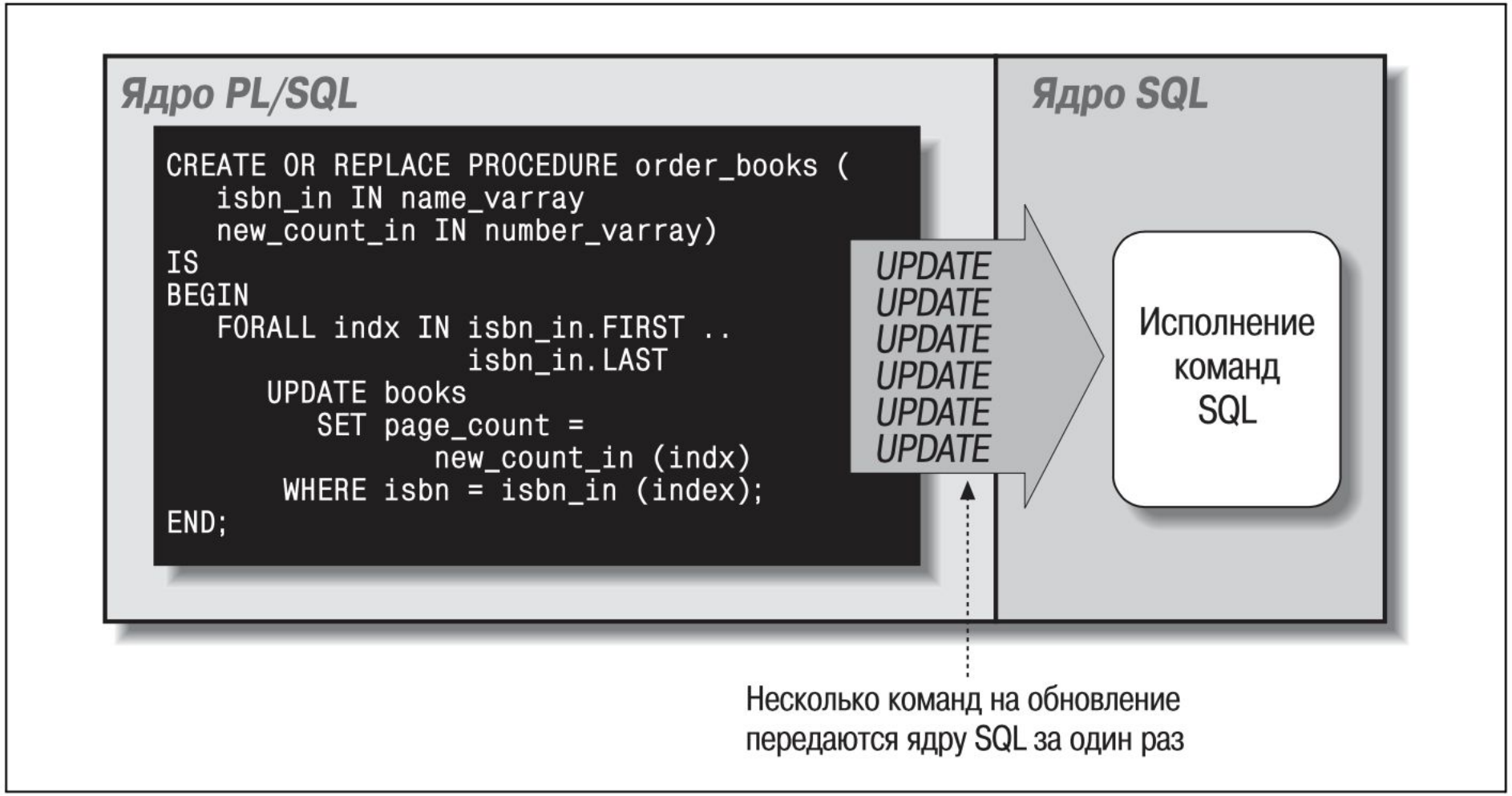

Рис. 1.4. Одно переключение контекста в операторе FORALL

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

```
FORALL индекс_строки IN
   [ нижняя_граница ... верхняя_граница |
     INDICES OF индексирующая_коллекция |
     VALUES OF индексирующая_коллекция
   ]
   [ SAVE EXCEPTIONS ]
   sql_команда;
```

где:

индекс_строки

Определяет коллекцию, элементы которой будет перебирать оператор FORALL.

нижняя_граница

Начальное значение индекса (строки или элемента коллекции) для выполнения операции.

верхняя_граница

Конечное значение индекса (строки или элемента коллекции) для выполнения операции.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

sql_команда

Команда SQL, которая должна быть выполнена для каждого элемента коллекции.

индексирующая_коллекция

Коллекция PL/SQL, используемая для выбора индексов в связанном массиве, заданном в *sql_команде*. Возможность использования операторов INDICES_OF и VALUES_OF появилась в версии Oracle Database 10*g*.

SAVE EXCEPTIONS

Необязательное предложение, которое сообщает оператору FORALL о необходимости обработки всех строк с сохранением всех возникающих исключений.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

При работе с FORALL необходимо соблюдать следующие правила:

- Телом оператора FORALL должна являться только одна команда DML: INSERT, UPDATE или DELETE.
- Оператор DML должен ссылаться на элементы коллекции посредством переменной *индекс_строки*, которая задана в операторе FORALL. Область действия переменной *индекс_строки* ограничивается оператором FORALL; вы не можете ссылаться на нее извне этого оператора. Однако обратите внимание, что верхняя и нижняя границы коллекций не обязательно должны охватывать все содержимое коллекций.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

- **Не следует объявлять переменную *индекс_строки*. Она объявляется неявно как переменная типа PLS_INTEGER ядром PL/SQL.**
- **Верхняя и нижняя границы должны задавать допустимый диапазон последовательных номеров индексов для коллекции (коллекций), заданных в команде SQL. Для разреженной коллекции будет сгенерирована следующая ошибка:**

```
ORA-22160: element at index [3] does not exist
```

Пример такой ситуации приведен в файле *diffcount.sql*, который можно найти на веб-сайте книги.

Тем не менее Oracle Database 10*g* поддерживает операторы INDICES OF и VALUES OF для разреженных коллекций (в которых пропущены какие-то элементы).

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

- **В операторе DML нельзя ссылаться на отдельные поля коллекций записей. Даже в случае, когда поле коллекции является коллекцией скаляров или коллекцией более сложных объектов, разрешено ссылаться только на строку коллекции целиком. Например, такой код:**

```
DECLARE
   TYPE employee_aat IS TABLE OF employee%ROWTYPE
      INDEX BY PLS_INTEGER;
   l_employees   employee_aat;
BEGIN
   FORALL l_index IN l_employees.FIRST .. l_employees.LAST
      INSERT INTO employee (employee_id, last_name)
        VALUES (l_employees (l_index).employee_id
              , l_employees (l_index).last_name
        );
END;
```

вызовет при компиляции такую ошибку:

```
PLS-00436: implementation restriction: cannot reference fields of BULK
In-BIND table of records
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

- **Индекс элемента коллекции, заданной в операторе DML, не может быть выражением. Например, выполнение следующего фрагмента**

```
DECLARE
   names name_varray := name_varray ();
BEGIN
   FORALL indx IN names.FIRST .. names.LAST
      DELETE FROM emp WHERE ename = names(indx+10);
END;
```

вызовет появление такой ошибки:

```
PLS-00430: FORALL iteration variable INDX is not allowed in this context
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

- **Используем предложение RETURNING в операторе FORALL для извлечения информации о каждой отдельной команде DELETE. Следует помнить, что предложение RETURNING оператора FORALL должно использовать вложенное предложение BULK COLLECT INTO («пакетная» операция для запросов):**

```
CREATE OR REPLACE FUNCTION remove_emps_by_dept (deptlist dlist_t)
   RETURN enolist_t
IS
   enolist enolist_t;
BEGIN
   FORALL aDept IN deptlist.FIRST..deptlist.LAST
      DELETE FROM emp WHERE deptno IN deptlist(aDept)
         RETURNING empno BULK COLLECT INTO enolist;
   RETURN enolist;
END;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции DML и FORALL

- **Используем индексы, определенные в одной коллекции, для определения того, какие строки из массива связывания (коллекции, заданной внутри команды SQL) должны использоваться динамическим оператором** INSERT**.**

```
FORALL indx IN INDICES OF l_top_employees
   EXECUTE IMMEDIATE
      'INSERT INTO ' || l_table || ' VALUES (:emp_pky, :new_salary)
      USING l_new_salaries(indx).employee_id,
            l_new_salaries(indx).salary;
```

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции BULK COLLECT

```
... BULK COLLECT INTO имя_коллекции[, имя_коллекции] ...
```

где *имя_коллекции* — параметр, определяющий коллекцию. При работе с BULK COLLECT необходимо учитывать несколько правил и ограничений:

- В версиях, предшествующих Oracle9*i* Database, BULK COLLECT может использоваться только со статическим SQL. В Oracle9*i* Database и Oracle Database 10*g* BULK COLLECT может применяться как для статического, так и для динамического SQL.
- Ключевые слова BULK COLLECT могут быть использованы в любом из следующих предложений: SELECT INTO, FETCH INTO и RETURNING INTO.
- Коллекции, которые указываются в предложении BULK COLLECT, могут хранить только скалярные значения (строки, числа и даты). Другими словами, невозможно извлечение строки данных в запись, являющуюся элементом другой коллекции.
- Ядро SQL автоматически инициализирует и расширяет коллекции, которые задаются в предложении BULK COLLECT. Заполнение начинается с индекса 1, элементы вставляются последовательно (плотно), любые определенные ранее элементы перезаписываются.
- Использование пакетной выборки SELECT...BULK COLLECT в операторе FORALL недопустимо.
- В случае, если не возвращено ни одной строки, SELECT...BULK COLLECT *не* инициирует исключения NO_DATA_FOUND. Вам необходимо проверить содержимое коллекции на предмет наличия в ней данных.
- Операция BULK COLLECT перед исполнением запроса делает пустой коллекцию, заданную в предложении INTO. Если запрос не возвращает строк, то метод COUNT этой коллекции будет возвращать 0.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

PL/SQL: пакетные операции BULK COLLECT

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.

Copyright © 1998 – 2020 Compass Plus Ltd. Public, v1.00r, 2020. Developed by: N.Raschupkina.