

Oracle Core

Тема 5

PL/SQL

Senior developer группы разработки Oracle Журавлев
Вячеслав

Senior developer группы разработки Oracle Юрченко Игорь

Содержание

- ☐ Знакомство
- ☐ Среда исполнения
- ☐ Структура PL/SQL блока
- ☐ Набор символов
- ☐ Основные типы и структуры данных (Null)
- ☐ Выражения
- ☐ Основные управляющие структуры

Знакомство

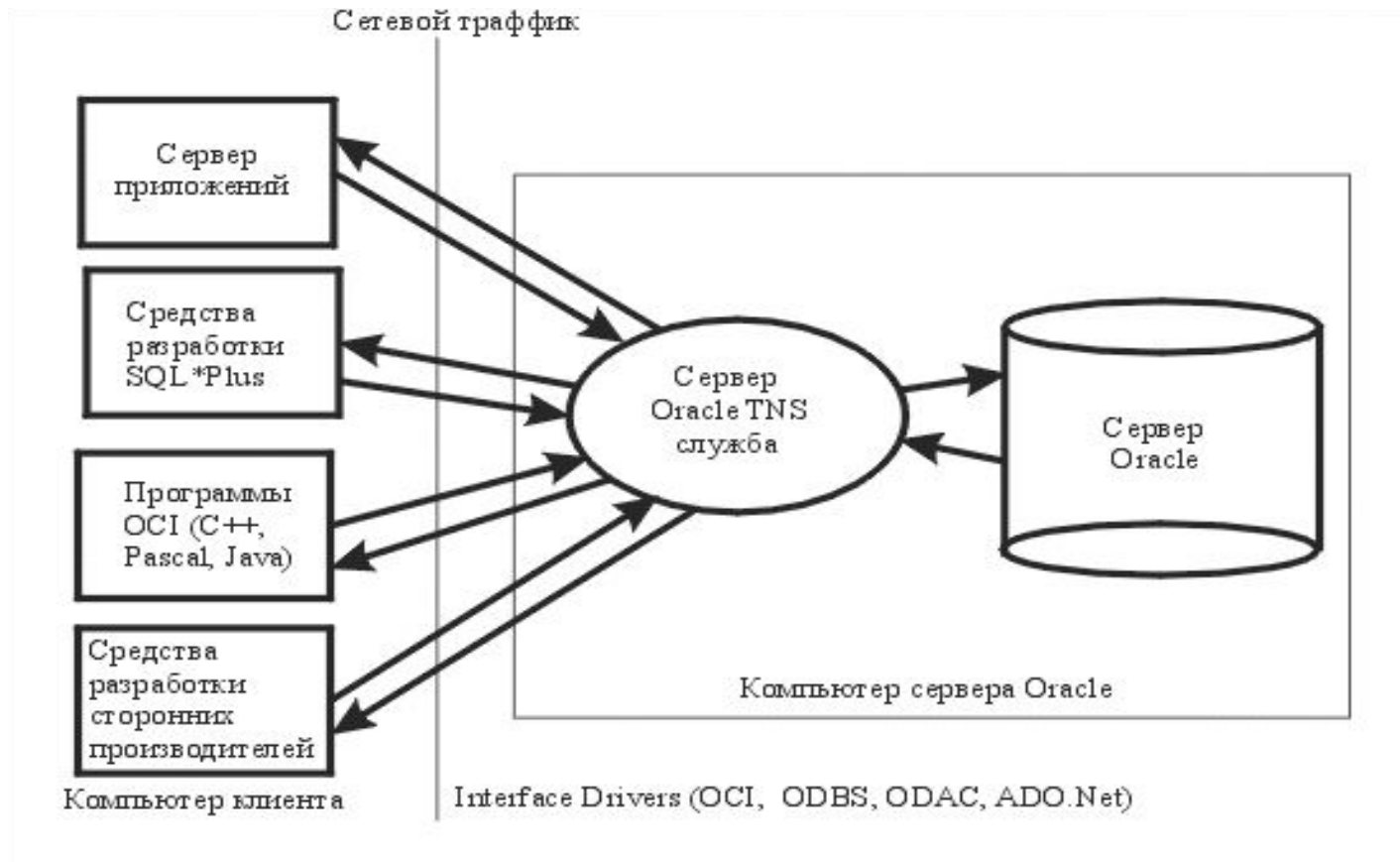
- ❑ Дата рождения - 1991г. В Oracle 6.0 – новый ключевой компонент – PL/SQL .
- ❑ PL/SQL – процедурный язык пошагового программирования, инкапсулирующий язык SQL. PL/SQL блочно ориентирован.
- ❑ PL/SQL – имеет строгие правила области видимости переменных, поддерживает параметризованные вызовы процедур и функций и так же унаследовал от языка ADA такое средство, как пакеты (package).
- ❑ PL/SQL – предусматривает строгий контроль типов, все ошибки несовместимости типов выявляются на этапе компиляции и выполнения. Так же поддерживается явное и неявное преобразование типов.
- ❑ Не является объектно-ориентированным, хотя имеет некоторые средства для создания и работы с объектами БД на уровне ООП.

Знакомство

- ❑ PL/SQL - является машинно независимым языком программирования.
- ❑ PL/SQL - поддерживает стандартные интерфейсы работы с языками высокого уровня такими как C, C++ - через предкомпиляторы поставляемые фирмой Oracle. (OCI - Oracle Call Interface) .
- ❑ Есть встроенные средства для работ с Internet и файловой системой.

Среда исполнения

Типичная клиент/серверная среда – узкое место – сеть



Среда исполнения

Исполнитель PL/SQL - компонент БД Oracle

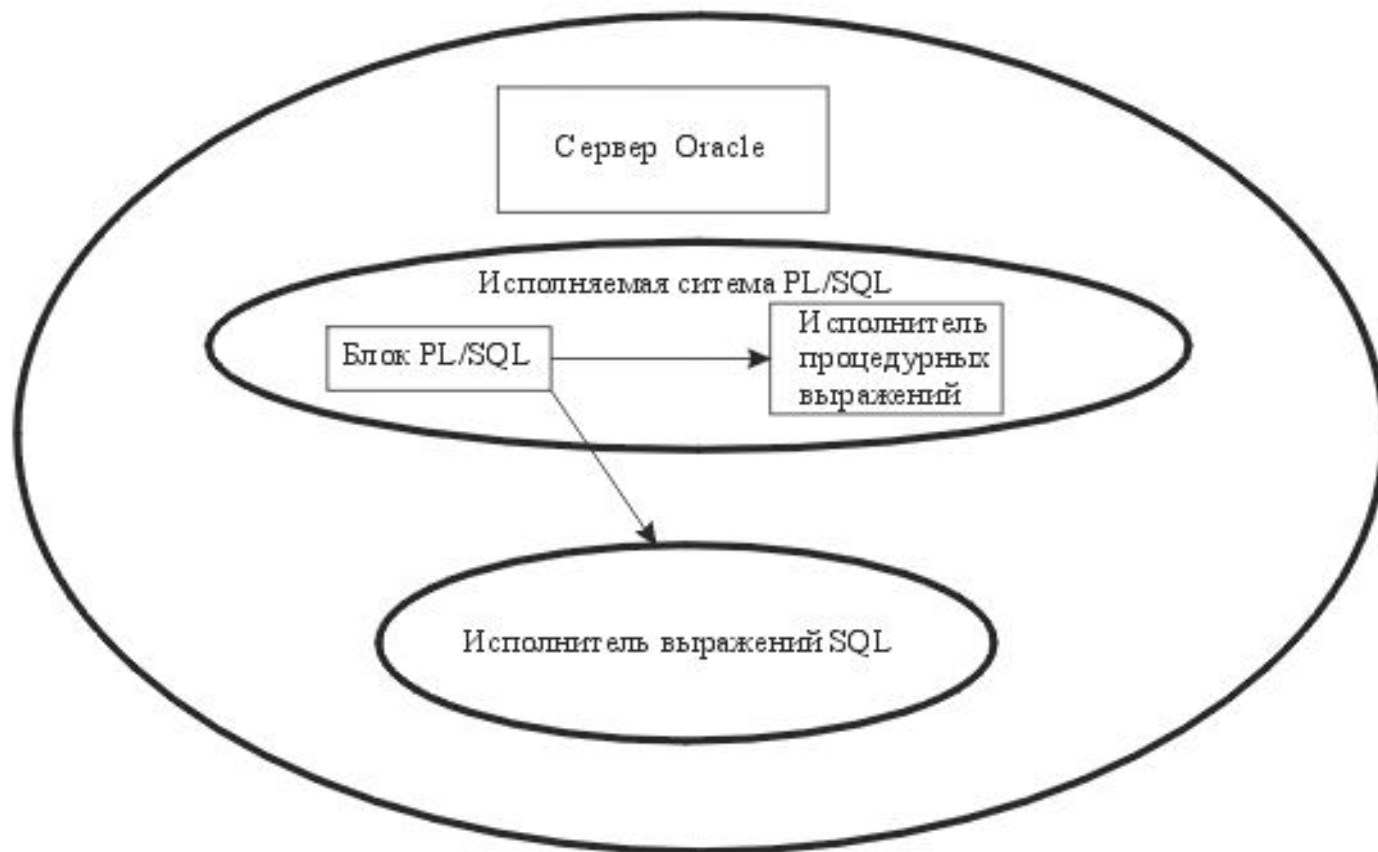
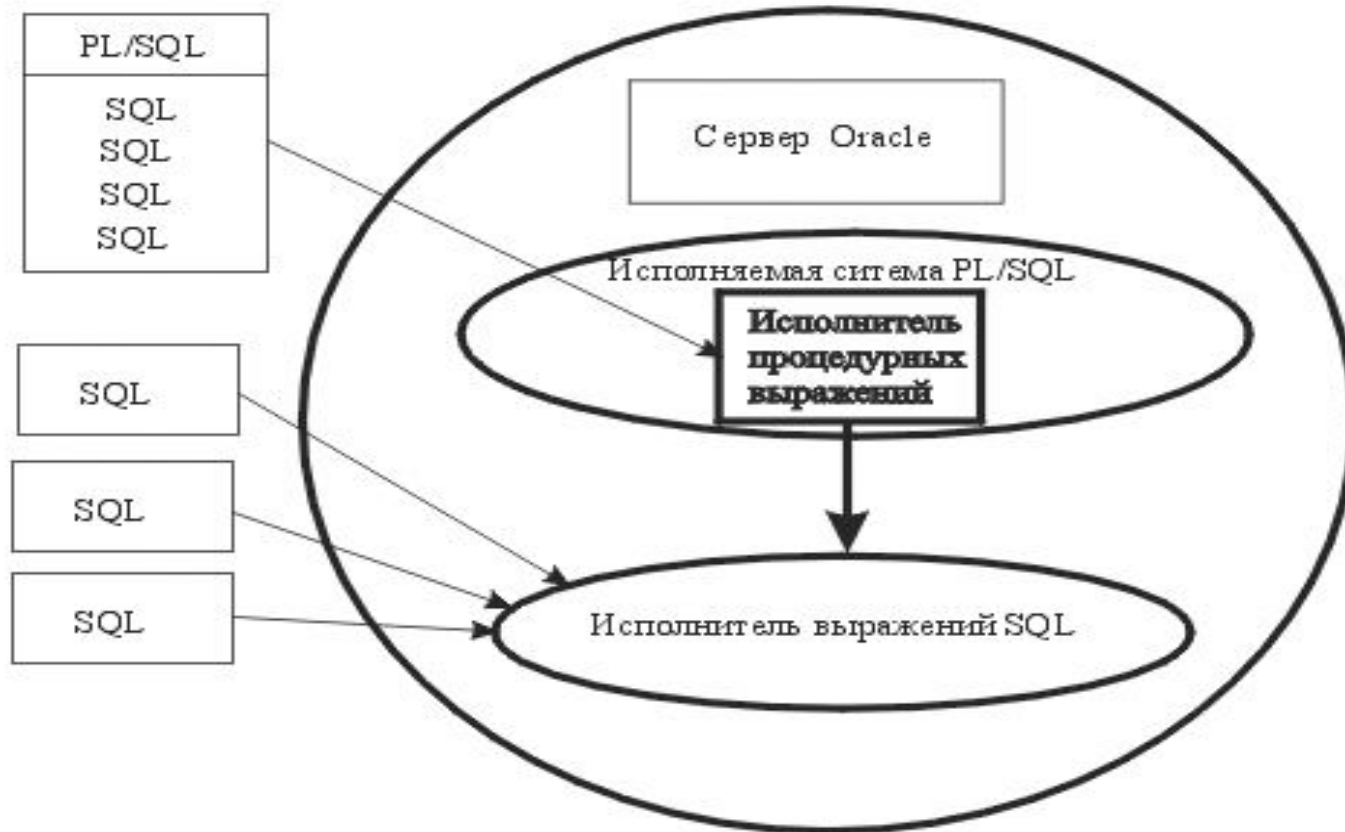


Рис. 2 Система исполнитель языка PL/SQL (является компонентом сервера БД Oracle)

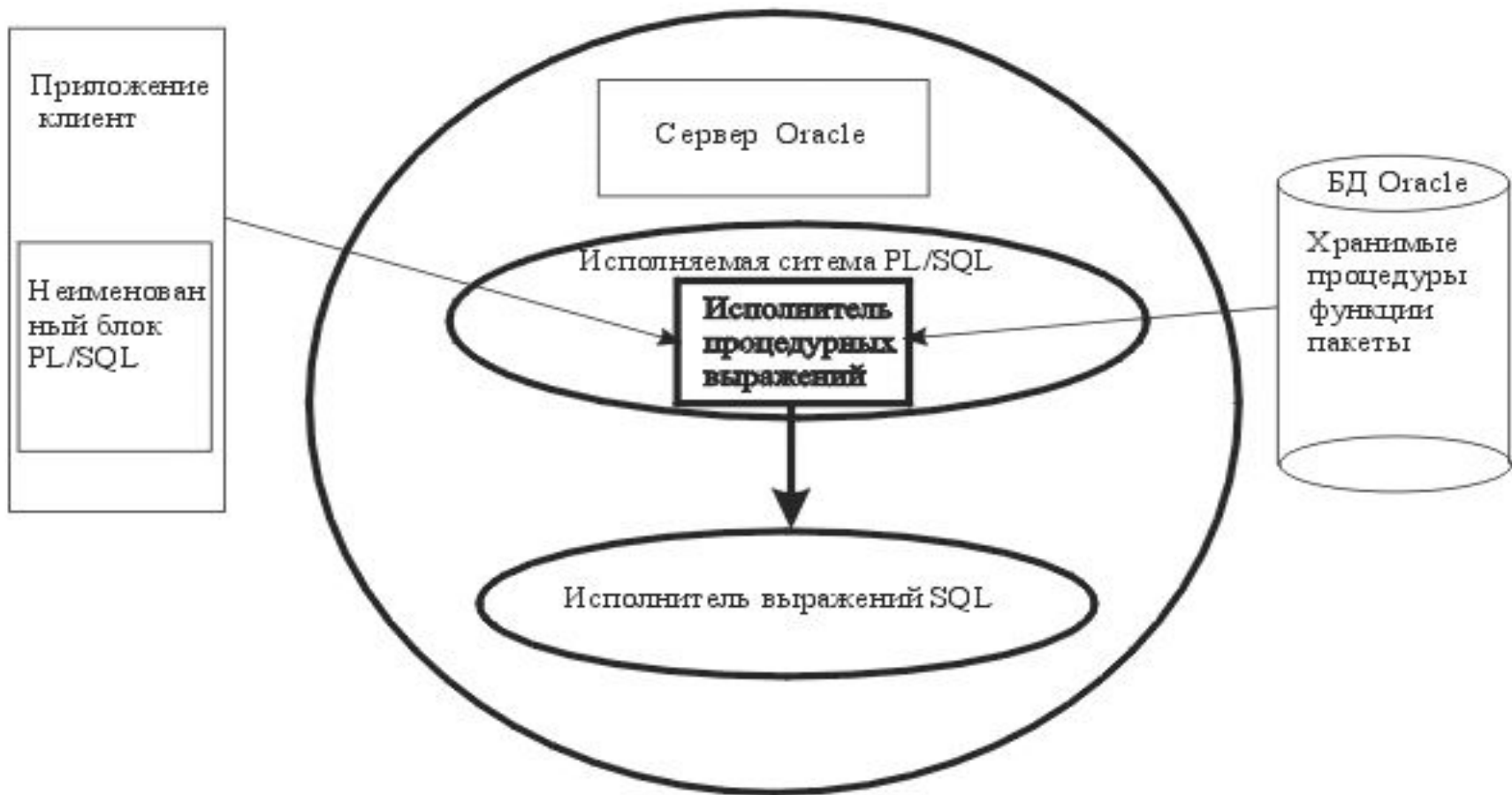
Среда исполнения

Группировка SQL кода в единый блок PL/SQL, сокращает нагрузку на сеть

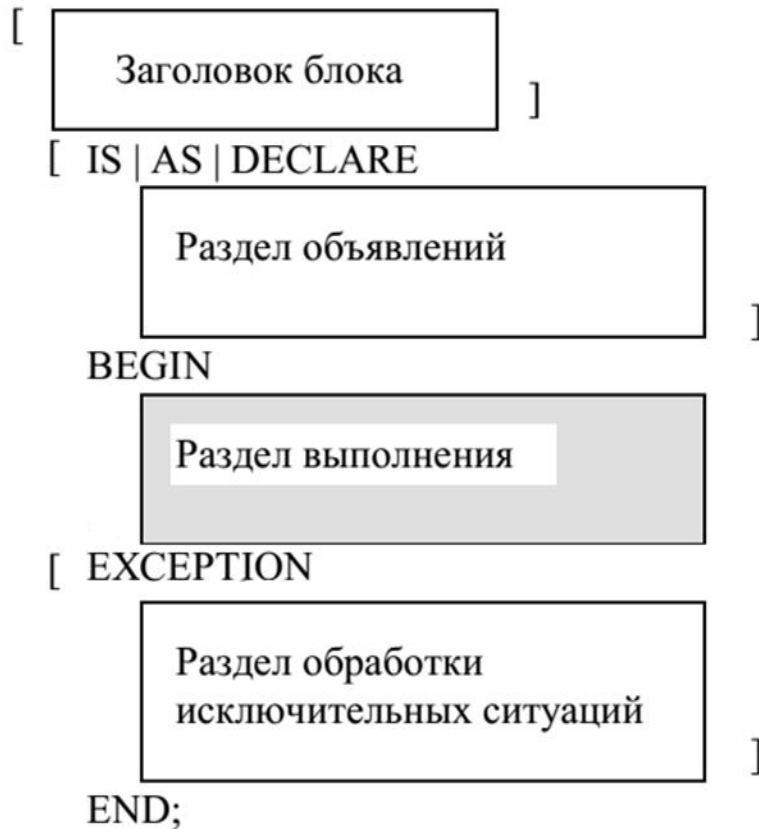


Среда исполнения

Вызов именованных хранимых процедур.



Структура PL/SQL блока



```
CREATE OR REPLACE PROCEDURE helloworld IS
```

```
    tmp_variable NUMBER;
```

```
BEGIN
```

```
    tmp_variable := 0;
```

```
EXCEPTION
```

```
    WHEN OTHERS THEN
```

```
        tmp_variable := 1;
```

```
END;
```

Структура PL/SQL блока

```
SQL> DECLARE
```

```
2      i                      NUMBER(2);  
3      string_variable VARCHAR2(20);
```

```
4 BEGIN
```

```
5     BEGIN
```

```
6         NULL; -- пустой оператор
```

```
7     END;
```

```
8     string_variable := 'Ave, Oracle'; -- присваиваем переменной значение
```

```
9
```

```
10    BEGIN
```

```
11        i := 20 / 0;
```

```
12    EXCEPTION
```

```
13        WHEN zero_divide THEN
```

```
14            dbms_output.put_line('error division by zero 1');
```

```
15    END;
```

```
16 EXCEPTION
```

```
17     WHEN OTHERS THEN
```

```
18         dbms_output.put_line('other error ' || SQLCODE);
```

```
19 END;
```

```
20 /
```

```
error division by zero 1
```

Процедура PL/SQL успешно завершена.

Структура PL/SQL блока

Анонимный блок (anonymous block)

```
SQL> BEGIN
  2      dbms_output.put_line('main block');
  3      BEGIN
  4          dbms_output.put_line('inner block lvl 1');
  5      DECLARE
  6          i NUMBER(1);
  7      BEGIN
  8          dbms_output.put_line('inner block lvl 2');
  9      END;
 10  EXCEPTION
 11      WHEN OTHERS THEN
 12          dbms_output.put_line('exception block');
 13  END;
 14 END;
 15 /

main block
inner block lvl 1
inner block lvl 2
```

Набор символов

Идентификаторы

Требования:

```
DECLARE

    new_variable NUMBER;
    linvalid_ident NUMBER;      -- начинается с цифры
    $invalid_ident NUMBER;      -- начинается не с буквы
    invalid-ident NUMBER;       -- содержит -
    invalid_%_ident NUMBER;     -- содержит %
    invalid_ident_because_very_large_length NUMBER; -- длинное название > 30 байт
    invalid ident NUMBER;       -- содержит пробел
    valid_ident1 NUMBER;
    valid_ident# NUMBER;

BEGIN

    new_variable := 2;
    VALID_IDENT1 := 2;

END;
```

Набор символов

Именовани

Уточнения:

☐ Не чувствителен к регистру:

☐ В

DEC

BEG

END



Все забыли последний слайд

Набор символов

Арифметические
операторы:

Оператор	Значение
+	Сложение и унарный плюс
-	Вычитание и унарный минус
*	Умножение
/	Деление
**	Возведение в степень

Операторы
отношения:

Оператор	Значение
=	Равенство
<	Меньше
>	Больше
<>	Не равно
!=	Не равно (альтернатива)
~=	Не равно (альтернатива)
^=	Не равно (альтернатива)
<=	Меньше или равно
>=	Больше или равно

```
DECLARE
```

```
    i NUMBER;
```

```
BEGIN
```

```
    i := 2 + 2 - 2 / 2 * 2 ** 2;
```

```
END;
```

Набор символов

Комментарии и

МЕТКИ

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

```
BEGIN
```

```
    -- однострочный комментарий
```

```
    /*
```

```
    многострочный
```

```
    комментарий
```

```
    */
```

```
    <<label1>>
```

```
    NULL;
```

```
END;
```

Основные типы и структуры данных

Вид данных	Описание
Скалярный	Переменные, представляющие собой ровно одно значение (числовое, дату и т.д.)
Составной	Переменные, представляющие именованную группу значений (запись, объект, массив)
Ссылка	Ссылка на объект или курсор
LOB	Указание на массив большого размера

Основные типы и структуры данных

Познакомимся сначала с

- ❑ **Null** специальное значение – означает отсутствие данных, констатацию того факта, что значение неизвестно
- ❑ По умолчанию это значение принимают переменные всех типов данных, если явно не указано ограничение NOT NULL
- ❑ Null ни равен ничему, даже другому Null
- ❑ Для сравнения существует сравнение через «is» и только так.

<pre>SQL> BEGIN 2 IF NULL = 3 THEN 4 dbms_ 5 ELSE 6 dbms_ 7 END IF; 8 END; 9 / False Процедура PL/SQL</pre>	<pre>SQL> BEGIN 2 IF NULL IS NULL 3 THEN 4 dbms_output.put_line('True'); 5 ELSE 6 dbms_output.put_line('False'); 7 END IF; 8 END; 9 / True Процедура PL/SQL успешно завершена.</pre>	<pre>SQL> BEGIN 2 IF NULL IS NULL 3 THEN 4 dbms_output.put_line('True'); 5 ELSE 6 dbms_output.put_line('False'); 7 END IF; 8 END; 9 / True Процедура PL/SQL успешно завершена.</pre>
---	--	--

Основные типы и структуры данных

Скалярные типы:

Числовые

Исторически первым для Oracle числовым типом является NUMBER. Он существует в трех вариантах:

- ❑ NUMBER – для хранения чисел «самого общего вида»;
- ❑ NUMBER(n) – для хранения целых с максимальной точностью мантиссы n десятичных позиций;
- ❑ NUMBER(n, m) (в частности NUMBER(*, m)) – для хранения чисел «с фиксированной десятичной точкой» с максимальной точностью мантиссы n десятичных позиций, из них m до десятичной точки

Все варианты типа NUMBER преобразуются к соответствующей им форме при помещении в базу, а при выборке из базы интерпретируются в соответствии с типом столбца.

Основные типы и структуры данных

Скалярные типы:

Числовые

Основные predefined числовые типы в PL/SQL

Data Type	Data Description
PLS_INTEGER or BINARY_INTEGER (одно и тоже)	Знаковое целое диапазон значений -2 147 483 648 до 2 147 483 647, размещается в 32 битах
BINARY_FLOAT	Числа с одинарной точностью, соответствует формату IEEE 754 – формат с плавающей запятой (32 бит, от $\pm 2^{-149}$ до $\pm 2^{127} \cdot (2 - 2^{-23})$)
BINARY_DOUBLE	Числа двойной точности, соответствует формату IEEE 754- формат с плавающей запятой (64 бит, диапазон от $\pm 2^{-1074}$ до $\pm 2^{1023} \cdot (2 - 2^{-52})$)
NUMBER [(p [, s])]	<p>С фиксированной или плавающей запятой с абсолютным значением в диапазоне от 1E-130 до (но не включая) 1.0E126. Может содержать ноль.</p> <p>Precision – общее число значащих цифр (max – 38)</p> <p>Scale – количество цифр справа от запятой (от -84 до 127)</p>

Основные типы и структуры данных

Скалярные типы: Числовые

Подтипы PLS_INTEGER

Data Type	Data Description
NATURAL	Неотрицательное PLS_INTEGER
NATURALN	Неотрицательное PLS_INTEGER значение с NOT NULL ограничением
POSITIVE	Положительное PLS_INTEGER значение (начинается с 1)
POSITIVEN	Положительное PLS_INTEGER значение с NOT NULL ограничением
SIGNTYPE	PLS_INTEGER значение -1, 0, или 1 (полезно при программировании tri-state логики (три состояния))
SIMPLE_INTEGER	PLS_INTEGER значение с NOT NULL ограничением

Основные типы и структуры данных

Скалярные типы: Числовые

Подтипы BINARY_FLOAT/DOUBLE:

Data Type	Data Description
SIMPLE_FLOAT	BINARY_FLOAT значение с NOT NULL ограничением
SIMPLE_DOUBLE	BINARY_DOUBLE значение с NOT NULL ограничением

Потеря точности при неявном преобразовании

```
SQL> DECLARE
2     small_num_variable  NUMBER(3, 1) := 7.5;
3     big_number          NUMBER(10) := 0;
4     naturaln_variable   NATURALN := 0;
5
6 BEGIN
7     big_number := big_number + small_num_variable;
8     big_number := big_number + small_num_variable;
9     dbms_output.put_line('big_number = ' || big_number);
10 END;
11 /
big_number = 16
```

Основные типы и структуры данных

Скалярные типы:

Числовые

С Null числовые типы ведут себя логично, но не явно.
При сложении числа с неизвестным значением – будет неизвестное значение. Надо запомнить!

```
SQL> DECLARE
  2      num_variable NUMBER(2);
  3  BEGIN
  4      num_variable := 3;
  5      num_variable := num_variable + NULL;
  6      IF num_variable IS NULL
  7      THEN
  8          dbms_output.put_line('num_variable is null');
  9      END IF;
 10  END;
 11  /
num_variable is null
```

Процедура PL/SQL успешно завершена.

Основные типы и структуры данных

Скалярные типы: Строковые ТИПЫ

Data Type	Data Description
CHAR [(size [BYTE CHAR])]	Строки фиксированной длины до 32767 байт (в Oracle SQL предел в 4000 байт)
<pre>SQL> DECLARE 2 char_variable CHAR(10); 3 BEGIN 4 char_variable := '3'; 5 dbms_output.put_line('char_variable = ' char_variable ');'); 6 END; 7 / char_variable = 3 ; Процедура PL/SQL успешно завершена.</pre>	
LONG RAW	Байтовая строка переменной длины до 32767 байт. Тип сохранен для обратной совместимости версий Oracle

Основные типы и структуры данных

Скалярные типы: Строковые типы

Эти типы формально можно причислить к строковым, но используются они для представления физических адресов данных:

Data Type	Data Description
ROWID	Двоичный массив фиксированной длины для хранения физического адреса данных Oracle в шестнадцатеричном формате OOOOOOFFFB BBBBRRR
UROWID [(size)]	«Универсальный» формат для ROWID: шестнадцатеричная строка переменной длины (до 4000 байт) с логическим значением ROWID. Используется для хранения адресов строк в индексно организованных (index organized) таблицах или в таблицах DB2 (через шлюз)

Основные типы и структуры данных

Скалярные типы: типы

Операции со стро

☐ Строки не скл

```
SQL> DECLARE
2      str_1 VARCHAR2(2);
3      str_2 VARCHAR2(2);
4      str_3 VARCHAR2(3);
5  BEGIN
6      str_1 := '15';
```

ируются

ИД

ВО

ПО

☐ Ср

ка

☐ Ср

ЗН

☐ Пр

пр

```
SQL> DECLARE
SQL> DECLARE
2      last_name1 VARCHAR(5) := 'ROW';
3      last_name2 CHAR(5) := 'ROW';
4  BEGIN
5      IF last_name1 = last_name2
6      THEN
7          dbms_output.put_line(last_name1||' is equal to '||last_name2||');
8      ELSE
9          dbms_output.put_line(last_name1||' is not equal to '||last_name2||');
10     END IF;
11 END;
12 /
ROW is not equal to ROW ;

Процедура PL/SQL успешно завершена.
```

») не

М не

Процедура PL/SQL успешно завершена.

Основные типы и структуры данных

Скалярные с NULL

❑ Пустая

❑ Length

❑ Сравн

недоп

❑ При к

```
SQL> DECLARE
```

```
2     str_1 VARCHAR2(4);
```

```
3     str_2 VARCHAR2(11);
```

```
4 BEGIN
```

```
5     str_1 := '';
```

```
6     IF str_1 IS NULL
```

```
7     THEN
```

```
8         dbms_output.put_line('str_1 IS NULL');
```

```
9     END IF;
```

```
10    IF length(str_1) IS NULL
```

```
11    THEN
```

```
12        dbms_output.put_line('length str_1 IS NULL');
```

```
13    END IF;
```

```
14
```

```
15    str_2 := 'Ave, Oracle' || str_1;
```

```
16    dbms_output.put_line('str_2 = ' || str_2);
```

```
17 END;
```

```
18 /
```

```
str_1 IS NULL
```

```
length str_1 IS NULL
```

```
str_2 = Ave, Oracle
```

```
Процедура PL/SQL успешно завершена.
```

МИ

Основные типы и структуры данных

Скалярные типы: Типы для моментов и интервалов

Data Type	Data Description
DATE	Допустимый диапазон дат от 1 января 4712 г. до н.э., до 31 декабря 9999 года нашей эры. Формат по умолчанию определяется в явном виде в параметре NLS_DATE_FORMAT или неявно в NLS_TERRITORY. Размер фиксируется в 7 байт. Этот тип данных содержит поля даты (Год, месяц, день, час, минуту и секунду. Не содержит дробной части секунд и часового пояса
TIMESTAMP [[fractional_seconds_precision]] [WITH TIME ZONE WITH LOCAL TIME ZONE]	Тоже что и дата + секунды имеют дробный формат. fractional_seconds_precision [0;9] – количество цифр в дробной части. По умолчанию 6. Может содержать часовой пояс или указать сразу локальный
INTERVAL YEAR [(year_precision)] TO MONTH	Хранит период времени в годах и месяцах. - year_precision - это количество цифр в YEAR. Допустимые значения от 0 до 9. По умолчанию 2.
INTERVAL DAY [(day_precision)] TO SECOND [[fractional_seconds_precision]]	Хранит период времени в днях и секундах. - day_precision—это кол-во цифр в DAY. Допустимые значения от 0 до 9. По умолчанию 2. - fractional_seconds_precision – количество цифр в дробной части SECOND. Допустимые значения от 0 до 9. Значение по умолчанию 6.

SQL> DECLARE

2 v_date DATE;

3 v_interval_year_month INTERVAL YEAR(2) TO MONTH;

4 v_interval_day_second INTERVAL DAY(3) TO SECOND;

5 BEGIN

6 v_date := '06.02.1985 12:00:05';

7 v_interval_year_month := '05-6';

8 v_interval_year_month := INTERVAL '3' MONTH;

9

10 v_interval_day_second := '200 06:02:00';

11 v_interval_day_second := INTERVAL '10' minute + INTERVAL '3' SECOND;

12

13 dbms_output.put_line('v_date = ' || v_date);

14 dbms_output.put_line('v_interval_year_month = ' || v_interval_year_month);

15 dbms_output.put_line('v_interval_day_second = ' || v_interval_day_second);

16

17 v_date := v_date + INTERVAL '29' YEAR + INTERVAL '8' MONTH + INTERVAL '11' DAY;

18 dbms_output.put_line('v_date = ' || v_date);

19 END;

20 /

v_date = 06.02.1985 12:00:05

v_interval_year_month = 05-06

v_interval_day_second = 200 06:02:00

v_date = 17.10.2014 12:00:05

Процедура PL/SQL успешно завершена.

Основные типы и структуры данных

Скалярные типы: Типы для моментов и интервалов

времени

Операции над
типами:

Operand 1	Operator	Operand 2	Result Type
datetime	+	interval	datetime
datetime	-	interval	datetime
interval	+	datetime	datetime
datetime	-	datetime	interval
interval	+	interval	interval
interval	-	interval	interval
interval	*	numeric	interval
numeric	*	interval	interval
interval	/	numeric	interval

Основные типы и структуры данных

Скалярные типы: Булевы типы

- ❑ тип для трехзначных переменных с допустимыми значениями TRUE, FALSE и NULL
- ❑ В выражениях с типом Boolean допускаются только булевы операнды

```
SQL> DECLARE
  2      v_boolean BOOLEAN;
  3  BEGIN
  4      v_boolean := TRUE AND FALSE;
  5  END;
  6  /
```

Процедура PL/SQL успешно завершена.

Основные типы и структуры данных

Типы LOB («большие неструктурированные объекты»)

- ❑ Позволяют хранить не сами данные, а «локаторы» (указатели) на данные, размещенные либо вне либо внутри БД

Data Type	Data Description
BFILE	Указатель на файл с данными в операционной системе (Не больше 4гб)
BLOB	Указатель на большой неструктурированный массив в БД (до 128 терабайт)
CLOB	Указатель на большой символьный массив в БД (до 128 терабайт)
NCLOB	Указатель на большой символьный массив в многобайтовой кодировке (до 128 терабайт)

Основные типы и структуры данных

Объявление переменных и постоянных

имя_переменной [CONSTANT] тип_данных [NOT NULL] [{:= | DEFAULT} выражение] ;

- ❑ Все переменные должны быть описаны в разделе объявления переменных;
- ❑ Указание CONSTANT задает неизменяемую константу.
- ❑ Выражение DEFAULT (эквивалент «:=») задает начальное значение.
- ❑ В выражении DEFAULT можно ссылаться на выше описанные переменные, если им на этот момент уже что-то присвоено значение по умолчанию

```
surname    VARCHAR2 (40) ;  
detector   BOOLEAN NOT NULL DEFAULT FALSE;
```


Составные типы

Записи

(Record)

В PL/SQL бывают трех видов:

- ☐ воспроизводящими структуру таблицы в БД;
- ☐ воспроизводящими структуру курсора в программе;
- ☐ задаваемыми пользователем произвольно

CREATE TYPE имя_типа IS RECORD (объявление переменных через “,”);

Особенности типа:

- ☐ Записи могут объявляться в разделе объявлений блока или в разделе глобального описания пакета
- ☐ Записи, повторяющие структуру таблицы или курсора, объявляются с помощью атрибута %ROWTYPE
- ☐ Записи, задаваемые пользователем, объявляются через предложение TYPE (без CREATE)
- ☐ Записи могут содержать другие записи
- ☐ Записи как целое не могут сравниваться логическими операциями (=, <> и пр.).
- ☐ Но индивидуальные поля указанные через точку, могут выставляться, читаться и сравниваться самостоятельно

Составные типы

Записи (Record) пример

создания:

```
SQL> DECLARE
```

```
2     TYPE t_department IS RECORD(  
3         title      VARCHAR2(30),  
4         position   VARCHAR2(100),  
5         customer   customers%ROWTYPE,  
6         empno      emp.empno%TYPE NOT NULL DEFAULT 100);  
7     v_department t_department;  
8
```

```
9 BEGIN
```

```
10    v_department.title      := 'MAIN DEP';  
11    v_department.customer.custno := 1;  
12    dbms_output.put_line('customer.custno = ' || v_department.customer.custno);  
13    dbms_output.put_line('empno = ' || v_department.empno);
```

```
14 END;
```

```
15 /
```

```
customer.custno = 1
```

```
empno = 100
```

```
Процедура PL/SQL успешно завершена.
```

Присвоить запись целиком возможно:

- ☐ оператором := имя_однотипной_записи
- ☐ конструкцией SELECT... INTO имя_записи FROM...
- ☐ конструкцией FETCH имя_курсора INTO{имя_записи |

Составные типы

Пользовательские подтипы:

SUBTYPE имя_подтипа IS базовый_тип [(ограничения)] [NOT NULL];

- ❑ базовый_тип – любой скалярный или пользовательский PL/SQL тип данных.
- ❑ Ограничения – здесь можно указать точность, размер, масштаб и т.д.

```
SQL> DECLARE
2     SUBTYPE birthdate IS DATE NOT NULL; -- основан на типе DATE с ограничением Not Null
3     SUBTYPE counter IS NATURAL; -- основан на подтипе NATURAL
4     TYPE timerec IS RECORD(
5         minutes INTEGER,
6         hours   INTEGER);
7     SUBTYPE finishtime IS timerec; -- основан на типе RECORD
8     SUBTYPE id_num IS customers.custno%TYPE; -- основан на типе столбца
9     SUBTYPE salary IS NATURAL RANGE 50 .. 100; -- основан на подтипе и ограничен в значении
10 BEGIN
11     NULL;
12 END;
13 /
```

Выражения

```
SQL> DECLARE
2     i NUMBER;
3     n CONSTANT VARCHAR2(255) := 'Scott';
4     c BOOLEAN;
5 BEGIN
6     i := sin(3) / cos(3);
7     dbms_output.put_line('Tangens of 3 radians: ' || to_char(i));
8     -- неявное преобразование возможно, но не рекомендуется:
9     dbms_output.put_line('The same tangens: ' || i);
10    c := n LIKE 'S%';
11    IF c
12    THEN
13        dbms_output.put_line(n || ' begins with S');
14    END IF;
15    dbms_output.put_line('User ' ||
16        CASE initcap(USER)
17            WHEN n THEN 'is SCOTT'
18            WHEN 'Scott' THEN 'is SCOTT too'
19            WHEN initcap(USER) THEN 'is SCOTT again'
20            ELSE 'SCOTT is not here'
21        END);
22
23    --- не рекомендуемое преобразование, зависит от локальных настроек
24    IF TIMESTAMP '2015-04-14 15:16:17' > systimestamp
25    THEN
26        dbms_output.put_line('ok');
27    END IF;
28 END;
29 /
```

Tangens of 3 radians: -,1425465430742778052956354105339134932514

The same tangens: -,1425465430742778052956354105339134932514

Scott begins with S

User is SCOTT again

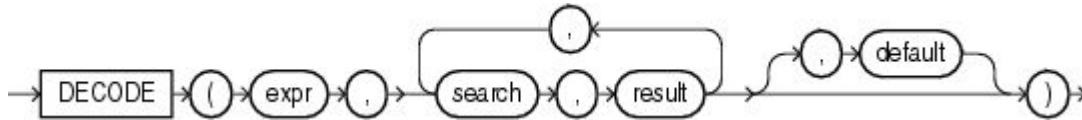
ok

Процедура PL/SQL успешно завершена.

ФУНКЦИИ ДЛЯ РАБОТЫ С NULL

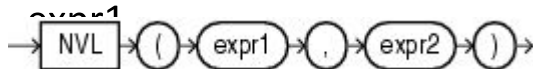
DECODE

Ищет первое совпадение `expr` и `search` и возвращает `result` в случае успеха, иначе возвращает `default` или `null`



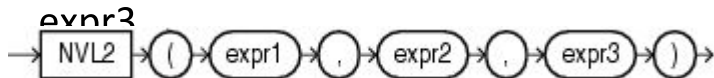
NVL

В случае если `expr1` is null тогда возвращается `expr2`, иначе



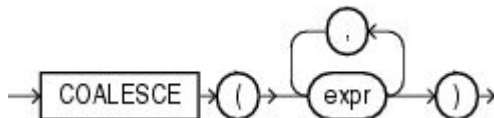
NVL2

В случае если `expr1` is null, тогда вернет `expr2` иначе



COALESCE

Возвращает первое не null



ПРЕОБРАЗОВАНИЕ ТИПОВ

Неявное и явное преобразование типов

Oracle рекомендует, чтобы Вы задавали явные преобразования, вместо того, чтобы полагаться на неявные или автоматические преобразования:

- ❑ Операторы SQL легче понять, когда Вы используете функции явного преобразования типа данных.
- ❑ Неявное преобразование типа данных может оказать отрицательное влияние на производительность, особенно если тип данных столбца преобразуется к типу данных константы, а не наоборот.
- ❑ Неявное преобразование зависит от контекста, в котором оно происходит и, возможно, не будет работать одинаково в каждом случае. Например, неявное преобразование значения типа данных VARCHAR2 может вернуть неожиданный год в зависимости от значения параметра NLS_DATE_FORMAT.
- ❑ Алгоритмы для неявного преобразования подвержены изменениям при обновлении версий продуктов Oracle. Поведение явных преобразований более предсказуемо.

ПРЕОБРАЗОВАНИЕ ТИПОВ

Неявное преобразование

ТИПОВ COLUMN	NUMBER	FLOAT	DOUBLE	DATE	TS	TS_LTZ	TS_TZ	CHAR	RAW
NUMBER	NATIVE	LEFT	LEFT	ERROR	ERROR	ERROR	ERROR	RIGHT	ERROR
FLOAT	RIGHT	NATIVE	LEFT	ERROR	ERROR	ERROR	ERROR	RIGHT	ERROR
DOUBLE	RIGHT	RIGHT	NATIVE	ERROR	ERROR	ERROR	ERROR	RIGHT	ERROR
DATE	ERROR	ERROR	ERROR	NATIVE	LEFT	LEFT	LEFT	RIGHT	ERROR
TS	ERROR	ERROR	ERROR	RIGHT	NATIVE	RIGHT	LEFT	RIGHT	ERROR
TS_LTZ	ERROR	ERROR	ERROR	RIGHT	RIGHT	NATIVE	LEFT	RIGHT	ERROR
TS_TZ	ERROR	ERROR	ERROR	RIGHT	RIGHT	RIGHT	NATIVE	RIGHT	ERROR
CHAR	LEFT	LEFT	LEFT	LEFT	LEFT	LEFT	LEFT	NATIVE	RIGHT
RAW	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	LEFT	NATIVE

NATIVE

Не конвертируется. “Native” (тип поля = типу литерала)

RIGHT

“Light” конвертация применяется к LITERAL, к типу поля или переменной(COLUMN) :
COLUMN = to_column_type(LITERAL).

LEFT

“Heavy” конвертация поля или переменной(COLUMN) к типу LITERAL type:
to_literal_type(COLUMN) = LITERAL.

ERROR

Неявное преобразование невозможно.

ПРЕОБРАЗОВАНИЕ ТИПОВ

Явное преобразование типов

from	to	VARCHAR2 NVARCHAR2	NUMBER	Datetime Interval	RAW	CLOB, NCLOB, BLOB	BINARY_FLOAT BINARY_DOUBLE
VARCHAR2 NVARCHAR2	TO_CHAR (char.) TO_NCHAR (char.)	TO_NUMBER	TO_DATE TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TO_DSINTERVAL	HEXTORAW	TO_CLOB TO_NCLOB	TO_BINARY_FLOAT TO_BINARY_DOUBLE	
NUMBER	TO_CHAR (number) TO_NCHAR (number)	--	TO_DATE NUMTOYM- INTERVAL NUMTODS- INTERVAL	--	--	TO_BINARY_FLOAT TO_BINARY_DOUBLE	
Datetime Interval	TO_CHAR (date) TO_NCHAR (datetime)	--	--	--	--	--	
RAW	RAWTOHEX RAWTONHEX	--	--	--	TO_BLOB	--	
CLOB, NCLOB, BLOB	TO_CHAR TO_NCHAR	--	--	--	TO_CLOB TO_NCLOB	--	
CLOB, NCLOB, BLOB	TO_CHAR TO_NCHAR	--	--	--	TO_CLOB TO_NCLOB	--	
BINARY_FLOAT BINARY DOUBLE	TO_CHAR (char.) TO NCHAR (char.)	TO_NUMBER	--	--	--	TO_BINARY_FLOAT TO BINARY DOUBLE	

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

PL/SQL имеет три категории операторов управления:

- ❑ Выбор по условию, запуск разного кода в зависимости от значения данных

Представляется конструкциями IF и CASE

- ❑ Выражения циклов, в них выполняются одни и те же операторы с серией различных значений данных.

Выражения циклов представлены конструкциями LOOP, FOR LOOP и WHILE LOOP.

Оператор EXIT передает управление в конец цикла.

Оператор CONTINUE завершает итерацию и передает управление следующей итерации. Указание этих операторов возможно задать через условие с помощью указания WHEN, после которого указывается необходимое условие выполнения оператора

- ❑ Выражения, которые управляют последовательностью вызова.

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выбор по условию: - конструкции IF

☐ Предложение

IF условное_выражение

THEN

 программный код в случае TRUE

END IF;

☐ Предложение

IF THEN ELSE

IF условное_выражение

THEN

 программный код в случае TRUE

ELSE

 программный код в случае FALSE/NULL

END IF;

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выбор по условию: - конструкции IF

❑ Предложение

```
IF условие_выражение1 THEN программный код1
ELSIF условие_выражение2 THEN программный код2
[ELSIF условие_выражениеi THEN программный кодi]...
[ELSE программный код в случае FALSE/NULL]
END IF;
```

```
SQL> BEGIN
  2     IF 1 = NULL
  3     THEN
  4         dbms_output.put_line('1 = null');
  5     ELSIF 1 != NULL
  6     THEN
  7         dbms_output.put_line('1 != null');
  8     ELSE
  9         dbms_output.put_line('null');
 10     END IF;
 11 END;
 12 /
null
```

Процедура PL/SQL успешно завершена.

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выбор по условию: - Конструкции CASE

❑ простой CASE

CASE выражение

WHEN выражение

[WHEN выражение

...

[ELSE программный код]

END CASE;

```
SQL> BEGIN
2      CASE
3          WHEN 1 = NULL THEN
4              dbms_output.put_line('1 = null');
5          WHEN 1 != NULL THEN
6              dbms_output.put_line('1 != null');
7          ELSE
8              dbms_output.put_line('null');
9      END CASE;
10 END;
11 /
null
```

Процедура PL/SQL успешно завершена.

❑ CASE с

CASE

WHEN условие THEN программный код

[WHEN условие THEN программный код] ...

[ELSE программный код]

END CASE;

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выражения циклов

Простейшие циклы представлены конструкцией:

```
[ метка ] LOOP  
    Программный код  
END LOOP [ метка ];
```

EXIT – прекращает выполнение цикла. WHEN – позволяет задать условие выхода

CONTINUE – завершает текущую итерацию цикла. WHEN – позволяет задать условие окончания итерации.

```
SQL> DECLARE  
2      i NUMBER(10);  
3  BEGIN  
4      i := 0;  
5      LOOP  
6          i := i + 1;  
7          CONTINUE WHEN i / 2 < 5;  
8          dbms_output.put_line('i = ' || i);  
9          EXIT WHEN i > 12;  
10     END LOOP main_loop;  
11 END;  
12 /  
i = 10  
i = 11  
i = 12  
i = 13
```

Процедура PL/SQL успешно завершена.

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выражения цикла

Цикл While

WHILE условие_вы

LOOP

программный код

END LOOP [слово-ко

Цикл FOR со

счетчиком

FOR индекс_цикла

верхнее_значение

LOOP

программный код

END LOOP [слово-ко

индекс_цикла заво
требуется.

указание **REVERSE** заставит перевернет счетчик и он будет декрементировать от **верхнее_значение** до **нижнее_значение**

```
SQL> DECLARE
2     i PLS_INTEGER;
3 BEGIN
4     i := 10;
5     FOR i IN 1 .. 10
6     LOOP
7         dbms_output.put_line('i = ' || i);
8         IF i > 3
9         THEN
10            EXIT;
11        END IF;
12    END LOOP;
13    dbms_output.put_line('i after cycle = ' ||
i);
14 END;
15 /
i = 1
i = 2
i = 3
i = 4
i after cycle = 10
```

Процедура PL/SQL успешно завершена.

влять его не

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Выражение

Цикл по курсору

FOR индекс

LOOP

программы

END LOOP

Объявление

и самостоятельное

Кроме того

☐ открывающая

☐ извлекающая

☐ курсора

```
SQL> BEGIN
  2      dbms_output.put_line('Employee in deptno = 3');
  3      dbms_output.put_line('-----');
  4      FOR cur IN (SELECT emp.empno,
  5                      emp.ename,
  6                      emp.sal,
  7                      emp.deptno
  8      FROM      emp
  9      WHERE     emp.deptno = 3)
 10      LOOP
 11          dbms_output.put_line('empno = ' || cur.empno);
 12          dbms_output.put_line('ename = ' || cur.ename);
 13          dbms_output.put_line('sal = ' || cur.sal);
 14          dbms_output.put_line('-----');
 15      END LOOP;
 16  END;
 17  /

Employee in deptno = 3
-----
empno = 1
ename = Bobovich
sal = 10
-----
empno = 2
ename = Perov
sal = 9,99
-----

Процедура PL/SQL успешно завершена.
```

мя_курсора%TYPE

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Управление последовательностью вызова (безусловный переход)

GOTO имя_метки выражение;

Ограничения

☐ нельзя пе

☐ нельзя пе

☐ нельзя пе

☐ нельзя пе

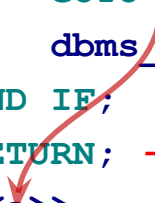
основной

☐ нельзя пе

исключит

Но передава

```
SQL> BEGIN
2      IF TRUE
3      THEN
4          GOTO a;
5          dbms_output.put_line('after goto');
6      END IF;
7      RETURN; -- принудительное завершение выполнения блока
8      <<a>>
9      dbms_output.put_line('ok');
10  END;
11  /
ok
Процедура PL/SQL успешно завершена.
```



ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

МЕТКИ В ЦИКЛАХ И БЛОКАХ

Для повышения надежности кода циклы можно размечать метками:

```
SQL> BEGIN
  2      <<year loop>>
  3      FOR YEAR IN 1998 .. 2003
  4      LOOP
  5          <<month_loop>>
  6          FOR MONTH IN 1 .. 12
  7          LOOP
  8              IF year_loop.year = 2000
  9                  AND MONTH = 1
 10              THEN
 11                  EXIT year_loop;
 12              END IF;
 13          END LOOP month_loop;
 14          dbms_output.put_line('Year ' || YEAR || ' gone');
 15      END LOOP year_loop;
 16      dbms_output.put_line('Year 2000 have come !');
 17  END;
 18  /
Year 1998 gone
Year 1999 gone
Year 2000 have come !
```

Процедура PL/SQL успешно завершена.

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ СТРУКТУРЫ

МЕТКИ В ЦИКЛАХ И БЛОКАХ

Этот же пример показывает, как метки можно использовать для организации доступа к переменным охватывающих их циклов. С этой же целью можно размечать метками вложенные блоки

```
SQL> BEGIN
  2      <<outer>>
  3      DECLARE
  4          i INTEGER := 1;
  5      BEGIN
  6          <<inner>>
  7          DECLARE
  8              i INTEGER := 2;
  9          BEGIN
 10              dbms_output.put_line('i = '||i); -- 1 или 2?
 11              dbms_output.put_line('outer.i = '||outer.i);
 12              dbms_output.put_line('inner.i = '||inner.i);
 13          END;
 14      END;
 15  END;
 16  /
i = 2
outer.i = 1
inner.i = 2
```

Процедура PL/SQL успешно завершена.

ССЫЛКИ

- ❑ http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/datatypes.htm#CIHBCHFH
- ❑ http://docs.oracle.com/cd/E11882_01/server.112/e26088/sql_elements001.htm#SQLRF30020
- ❑ http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/overview.htm#LNPLS001
- ❑ http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/create_type.htm#i2083561
- ❑ http://docs.oracle.com/cd/B10501_01/appdev.920/a96583/cci06met.htm
- ❑ http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/controlstatements.htm#LNPLS99972
- ❑ <http://habrahabr.ru/post/127327/>
- ❑ «Oracle PL/SQL ДЛЯ ПРОФЕССИОНАЛОВ», 3-Е ИЗДАНИЕ - С. Фейерштейн, Б. Прибыл
- ❑ «Введение в PL/SQL», В. В. Пржиялковский