

Отчет по лабораторной работе № 5 «Основы PL/SQL»			
дата	Оценка (max 5)	Бонус за сложность	подпись

Цели работы:Изучение операторов PL/SQL**Задачи работы:**

- создание ненаименованных блоков
- создание функции
- создание процедур
- создание пакетов
- создание триггеров

Задание повышенной сложности (бонус за сложность – 5 баллов):

- выполнение триггеров – фильтров входных данных строкового типа

Краткий конспект теоретической части (ответы на контрольные вопросы)

Определение «функции» _____

Определение «процедуры» _____

Определение «пакета» _____

Определение «триггера» _____

Структура ненаименованного PL/SQL блока _____

PL/SQL – язык Oracle четвертого поколения, объединяющий структурированные элементы процедурного языка программирования с языком SQL, разработанный специально для организации вычислений в среде клиент/сервер.

Свойства:

- Он позволяет передать на сервер программный блок PL/SQL, содержащий логику приложения, как оператор SQL, одним запросом.
- Используя PL/SQL, можно значительно уменьшить объем обработки в клиентской части приложения и нагрузку на сеть. Например, может понадобиться выполнить различные наборы операторов SQL в зависимости от результата некоторого запроса. Запрос, последующие операторы SQL и операторы условного управления могут быть включены в один блок PL/SQL и пересланы серверу за одно обращение к сети.
- Вся логика приложений делится на клиентскую и серверную части. Серверная часть может быть реализована в виде функций, хранимых процедур и пакетов.

Базовые элементы:

Ненаименованные блоки

Законченный логический блок, реализованный на PL/SQL, выполняемый в терминальном режиме, без возможности обращения к нему из других модулей.

Наименованные блоки:

Функции. Часть логики приложения ориентированной на выполнение конкретного комплекса операций на сервере, результат которых возвращается в виде значения функции. Откомпилированные функции и их исходные тексты содержатся в базе данных.

Хранимые процедуры. Часть логики приложения, особенно нуждающаяся в доступе к базе данных, может храниться там, где она обрабатывается (на сервере). Хранимые процедуры не возвращают значения результата, обеспечивают удобный и эффективный механизм безопасности. Откомпилированные хранимые процедуры и их исходные тексты содержатся в базе данных.

Пакеты. Часть логики приложений: функций и процедур, предназначенных для решения задач в рамках одного модуля (подсистемы) АИС.

Триггеры базы данных. Можно использовать триггеры, чтобы организовать сложный контроль целостности, выполнять протоколирование (аудит) и другие функции безопасности, реализовать в приложениях выдачу предупреждений и мониторинг.

Декларативная целостность. Ограничения активизируются сервером всякий раз, когда записи вставляются, обновляются или удаляются. В дополнение к ограничениям ссылочной целостности, которые проверяют соответствие первичного и внешнего ключей, можно также накладывать ограничения на значения, содержащиеся в столбцах таблицы.

Принципы разработки ненаименованных PL/SQL блоков

Программы на PL/SQL имеют блочную структуру:

DECLARE

-- объявления переменных, констант, типов данных, курсоров, функций и процедур.

BEGIN

-- выполняемый код

EXCEPTION

-- обработка исключений

END;



Пример 1:

Вывести в выходной поток SQL+ строковую переменную «HELLO WORLD»

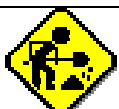
```
SET SERVEROUTPUT ON  -- определяет вывод SQL*Plus всю информацию
                      -- возвращаемую сервером.

BEGIN
    DBMS_OUTPUT.enable;          -- включение механизма вывода
    DBMS_OUTPUT.put_line('HELLO WORLD'); -- печать строки
END;
/ -- указание к выполнению блока PL/SQL
```

Вклейте результат работы программы в SQL+

```
HELLO WORLD
```

```
PL/SQL procedure successfully completed.
```



Пример 2:

Пример: Разработать ненаименованный PL/SQL блок, вычисляющий площадь круга и длину окружности.

```
DECLARE
PI CONSTANT REAL := 3.141519265359;
LOKR REAL;
SKRG REAL;
RADIUS REAL := &RADIUS; -- указывает на необходимость ввода перем.
BEGIN
    LOKR := PI * RADIUS * 2.0;
    SKRG := PI *RADIUS ** 2;
    DBMS_OUTPUT.put_line('Радиус = ' || to_char(RADIUS)|| ', ДЛИНА
ОКРУЖНОСТИ =' || to_char(LOKR) || ', ПЛОЩАДЬ КРУГА =' || to_char(SKRG));
END;
/
```

Вклейте результат работы программы в SQL+

```
Радиус = 1, ДЛИНА ОКРУЖНОСТИ =6.283038530718, ПЛОЩАДЬ КРУГА =3.141519265359
```

```
PL/SQL procedure successfully completed.
```

Управляющие структуры PL/SQL


Для управления работой PL/SQL блоков используются **условные операторы и циклы**.

Условные операторы: IF-THEN-ELSE


Синтаксис:

```
IF логическое выражение 1 THEN
    операторы 1;
    [ELSIF логическое выражение 2 THEN
        операторы 1;]
.....
ELSE операторы N;
END IF;
```

IF-THEN-ELSE

	Пример 3: Модифицировать пример, чтобы учитывать проверку на равенство NULL значения v_num1
<pre>DECLARE v_num1 NUMBER; v_num2 NUMBER; v_result VARCHAR(7); BEGIN IF v_num1 >= v_num2 THEN v_result := 'No'; ELSE v_result := 'Yes'; END IF; END;</pre>	<pre>1 DECLARE 2 v_num1 NUMBER; 3 v_num2 NUMBER; 4 v_result VARCHAR(32); 5 BEGIN 6 IF v_num1 IS NULL 7 OR v_num2 IS NULL THEN 8 v_result := 'error, v_num1 or v_num2 is null'; 9 ELSIF v_num1 >= v_num2 THEN 10 v_result := 'no'; 11 ELSE 12 v_result := 'yes'; 13 END IF; 14 dbms_output.put_line(to_char(v_result)); 15 END; 16 /</pre> <div>error, v_num1 or v_num2 is null</div>

Работа с циклами

	Пример 4: Привести примеры работы с циклами
Простые циклы: LOOP операторы; END LOOP;	<div><div>1 DECLARE 2 x NUMBER := 5; 3 BEGIN 4 dbms_output.enable(); 5 LOOP 6 dbms_output.put_line('Поставьте оценку'); 7 x := :x; 8 IF x = 5 THEN 9 EXIT; 10 END IF; 11 END LOOP; 12 WHILE x <> 5 LOOP 13 dbms_output.put_line('Поставьте оценку'); 14 x := :x; 15 END LOOP; 16 FOR i IN 0..10 LOOP 17 dbms_output.put_line(concat(18 'Поставьте оценку за семинар №' 19 ,to_char(i) 20)); 21 x := :x; 22 END LOOP; 23 END; 24 /</div><div>Числовые циклы: FOR счетчик IN min..max LOOP операторы; END LOOP;</div></div>



Пример 5:

Модифицировать пример, используя различные варианты задания циклов

```
1 DECLARE
2     dt     DATE;
3     radius REAL := &radius;
4 BEGIN
5     dt := current_timestamp;
6     LOOP
7         radius := radius + 1;
8         dbms_output.put_line(to_char(2 * 3.14 * (radius))
9                               || ' время '
10                              || to_char(
11                                  (current_timestamp - dt)
12                                  , 'SS'
13                              ));
14         IF radius > 5 THEN
15             EXIT;
16         END IF;
17     END LOOP;
18 END;
19 /
```

PL/SQL procedure successfully completed.

```
12.56 время +000000 00:00:00.725332000
18.84 время +000000 00:00:00.725487000
25.12 время +000000 00:00:00.725498000
31.4  время +000000 00:00:00.725505000
37.68 время +000000 00:00:00.725511000
```



Пример 6:

Приведите собственный пример обработки исключительных ситуаций

Prompt Обработка исключительных ситуаций

DECLARE

x real := &x;

n integer := 0;

BEGIN

dbms_output.enable;

dbms_output.put_line('Значения функции');

dbms_output.put_line('abs(x)= ' || to_char(ABS(x)));
n:=n+1;

dbms_output.put_line('ceil(x)= ' || to_char(ceil(x)));
n:=n+1;

exception

when others then begin

dbms_output.put_line('ERRORS');

if n=0 then

dbms_output.put_line('Ошибка в функции ABS(x)');

end;

END; /

DECLARE

degree INTEGER := °ree;

BEGIN

IF degree IN (1
 , 2
 , 3
 , 4) THEN

RAISE invalid_number;

END IF;

EXCEPTION

WHEN invalid_number THEN

dbms_output.put_line('оценка слишком мала');

ROLLBACK;

END;

/

PL/SQL procedure successfully completed.

оценка слишком мала

Библиотечные функции в Oracle

Числовые функции

Функция	Возвращаемое значение
ABS(n)	Абсолютное значение величины <i>n</i> .
CEIL(n)	Наименьшее целое, большее или равное <i>n</i> ,
COS(n)	Косинус <i>n</i> (угла, выраженного в радианах).
COSH(n)	Гиперболический косинус <i>n</i> .
EXP(n)	<i>e</i> в степени <i>n</i> .
FLOOR(n)	Наибольшее целое, меньшее или равное <i>n</i> .
LN(n)	Натуральный логарифм <i>n</i> , где <i>n</i> >0.
LOG(m,n)	Логарифм <i>n</i> по основанию <i>m</i> .
MOD(m,n)	Остаток от деления <i>m</i> на <i>n</i> .
POWER(m,n)	<i>m</i> в степени <i>n</i> .
ROUND(n[,m])	<i>n</i> , округленное до <i>m</i> позиций после десятичной точки. По умолчанию <i>m</i> равно нулю.
SIGN(n)	Если <i>n</i> <0, -1; если <i>n</i> =0, 0; если <i>n</i> >0, 1.
SIN(n)	Синус <i>n</i> (угла, выраженного в радианах).
SINH(n)	Гиперболический синус.
SQRT(n)	Квадратный корень от <i>n</i> , если <i>n</i> <0, возвращает значение NULL.
TAN(n)	Тангенс <i>n</i> (угла, выраженного в радианах).
TANH(n)	Гиперболический тангенс <i>n</i> .
TRUNC(n[,m])	<i>n</i> , усеченное до <i>m</i> позиций после от десятичной точки. По умолчанию <i>m</i> равно нулю.



Пример 7:

Приведите пример использования библиотечной функции данного типа

```

DECLARE
    degree INTEGER := &degree;
BEGIN
    dbms_output.put_line(ceil(degree));
END;
/

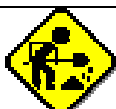
```

4

PL/SQL procedure successfully completed.

Символьные функции, возвращающие символьные значения:

Функция 1	Возвращаемое значение
CHR(n)	Символ с кодом <i>n</i> .
CONCAT(char1,char2)	Конкатенация символьных строк <i>char1</i> и <i>char2</i> .
INITCAP(char)	Символьная строка <i>char</i> , первые буквы всех слов в которой преобразованы в прописные.
LOWER(char)	Символьная строка <i>char</i> , все буквы которой преобразованы в строчные.
LPAD(char1.n [,char2])	Символьная строка <i>char1</i> , которая дополняется слева последовательностью символов из <i>char2</i> так, чтобы общая длина строки стала равна <i>n</i> . Значение <i>char2</i> по умолчанию - " (один пробел). Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
LTRIM(char[,set])	Символьная строка <i>char</i> , в которой удалены все символы от начала вплоть до первого символа, которого нет в строке <i>set</i> . Значение <i>set</i> по умолчанию - " (один пробел).
NLS_INITCAP(char[,nls_sort])	Символьная строка <i>char</i> , в которой первые буквы всех слов преобразованы в прописные. Параметр <i>nls_sort</i> определяет последовательность сортировки.
NLS_LOWER(char[,nls_sort])	Символьная строка <i>char</i> , все буквы которой преобразованы в строчные. Параметр <i>tils-sort</i> определяет последовательность сортировки.
NLS_UPPER(char[,nls_sort])	Символьная строка <i>char</i> , все буквы которой преобразованы в прописные. Параметр <i>nts_sort</i> определяет последовательность сортировки.
REPLACE(char, search_string [,replacement_string])	Символьная строка <i>char</i> , в которой все фрагменты <i>search_string</i> заменены на <i>replacement_string</i> . Если параметр <i>replacement_string</i> не определен, все фрагменты <i>search-string</i> удаляются.
RPAD(char1 [,char2])	Символьная строка <i>char1</i> , которая дополнена справа последовательностью символов из <i>char2</i> так, что общая длина строки равна <i>n</i> . Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
RTRIM(char[,set])	Символьная строка <i>char</i> , в которой удалены все символы справа вплоть до первого символа, которого нет в строке <i>set</i> . Значение параметра <i>set</i> по умолчанию - ¹ (один пробел).
SOUNDEX(char)	Символьная строка, содержащая фонетическое представление для <i>char</i> , на английском языке.
SUBSTR(char, m[,n])	Фрагмент символьной строки <i>char</i> , начинающийся с символа <i>m</i> , длиной <i>n</i> символов (до конца строки, если параметр <i>n</i> не указан).
SUBSTRB(char, m[,n])	Фрагмент символьной строки <i>char</i> , начинающийся с символа <i>m</i> , длиной <i>l</i> байтов (до конца строки, если параметр <i>n</i> не указан).
TRANSLATE(char, from, to)	Символьная строка <i>char</i> , в которой все символы, встречающиеся в строке <i>from</i> , заменены на соответствующие символы из <i>to</i> .
UPPER(char)	Символьная строка <i>char</i> , в которой все буквы преобразованы в прописные.



Пример 8:

Приведите пример использования библиотечной функции данного типа

```

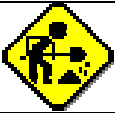
1  BEGIN
2  dbms_output.put_line(replace(
3      'text with some secret words like password etc'
4      , 'secret'
5      ));
6  END;
7  /

```

text with some words like password etc

Символьные функции, возвращающие числовые значения

Функция	Возвращаемое значение
ASCII(char)	Возвращает десятичный код первого символа строки <i>char</i> в кодировке, принятой в базе данных. (Код ASCII в системах, использующих кодировку ASCII). Возвращает значение первого байта многобайтового символа.
INSTR(char1, char2[, n[, m]])	Позиция первого символа <i>m</i> -ого фрагмента строки <i>char1</i> , совпадающего со строкой <i>char2</i> , начиная с <i>n</i> -ого символа. По умолчанию <i>n</i> и <i>m</i> равны 1. Номер символа отсчитывается от первого символа строки <i>char1</i> , даже когда <i>n</i> > 1.
INSTRB(char1, char2[, n[, m]])	Позиция первого символа <i>m</i> -ого фрагмента строки <i>char1</i> , совпадающего со строкой <i>char2</i> , начиная с <i>m</i> -ого байта. По умолчанию <i>n</i> и <i>m</i> равны 1. Номер байта отсчитывается от первого символа строки <i>char1</i> , даже когда <i>n</i> > 1.
LENGTH(char)	Длина строки <i>char</i> в символах.
LENGTHB(char)	Длина строки <i>char</i> в байтах.
NLSORT(char1, char2[, n[, m]])	Зависящее от национального языка значение, используемое при сортировке строки <i>char</i> .

	Пример 9: Приведите пример использования библиотечной функции данного типа
<pre>BEGIN dbms_output.put_line(length('text with some secret words like password etc')); END;</pre>	<pre>PL/SQL procedure successfully completed. 45</pre>

Групповые функции


Функция	Возвращаемое значение
AVG([DISTINCT ALL]n)	Среднее значение от <i>n</i> , нулевые значения опускаются.
COUNT([ALL]*)	Число строк, извлекаемых в запросе или подзапросе.
COUNT(DISTINCT ALL] expr)	Число строк, для которых <i>expr</i> принимает не пустое значение.
MAX([DISTINCT ALL] expr)	Максимальное значение выражения <i>expr</i> .
MIN((DISTINCT ALL] expr)	Минимальное значение выражения <i>expr</i> .
STDDEV([DISTINCT ALL] n)	Стандартное отклонение величины <i>n</i> , нулевые значения опускаются.
SUM([DISTINCT ALL] n)	Сумма значений <i>n</i>
VARIANCE([DISTINCT ALL]n)	Дисперсия величины <i>n</i> , нулевые значения опускаются.

Функции работы с датами

Функция	Возвращаемое значение
ADD-MONTHS (d,n)	Дата d плюс n месяцев.
LAST-DAY (d)	Последнее число месяца, указанного в d
MONTHS-BETWEEN (d1, d2)	Число месяцев между датами d1 и d2.
NEW-TIME (d, a, b)	Дата и время в часовом поясе a, соответствующие дате и времени в часовом поясе b, при этом d,a и b значения типа CHAR, определяющие часовые пояса.
NEW-DAY (d, char)	Дата первого после даты (/дня недели, название которого записано в <i>с1шг</i> .
SYSDATE	Текущая дата и время.

Усечение и округление дат

Функция	Возвращаемое значение
ROUND(d [,fmt])	Дата d, округленная до единиц, указанных в форматной маске.
TRUNC(d [,fmt])	Дата d, усеченная по форматной маске fmt.

	Пример 10: Приведите пример использования библиотечной функции данного типа				
<pre>SELECT trunc(sysdate) FROM dual;</pre>	<table><tr><th></th><th>TRUNC(SYSDATE)</th></tr><tr><td>1</td><td>22.04.25</td></tr></table>		TRUNC(SYSDATE)	1	22.04.25
	TRUNC(SYSDATE)				
1	22.04.25				

Форматные маски дат для функций ROUND и TRUNC.

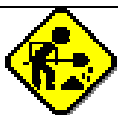
В таблице перечислены форматные маски, которые можно использовать в функциях ROUND и TRUNC. По умолчанию используется форматная маска “DD”.

Форматная маска	Возвращаемое значение
CC или SCC	Первый день столетия
SYYYYY или YYYYY или YYY или YY или Y или YEAR или SYEAR	Первый день года (округляется до 1 июля)
Q	Первый день квартала (округляется до 16 числа второго месяца квартала)
MONTH или MON или MM или RM	Первый день месяца (округляется до 16 числа)
WW или IW	Тот же день недели, что и первый день текущего года
W	Тот же день недели, что и первый день текущего месяца
DDD или DDD или J	День
DAY или DY или D	Первый день недели
HH HH12 HH24	Час
MI	Минута

Форматные маски дат в TO_CHAR и TO_DATE.

Элементы форматной маски даты перечислены в приведенной ниже таблице. Любую комбинацию этих элементов можно использовать как аргумент fmt функций TO_CHAR или TO_DATE. По умолчанию fmt равен 'DD-MON-YY'.

Элемент формата	Возвращаемое значение
SCC или CC	Столетие; если указано 'S' то перед датами до нашей эры ставится '-'.
YYYY или SYYYY	Год; если указано 'S' то перед датами до нашей эры ставится '-'. YYY или YY или Y] Последние 3, 2, или 1 цифра года.
IYYY	4 цифры года по стандарту ISO. IYY или IY или I] Последние 3, 2, или 1 цифра года по стандарту ISO.
Y,YYY	Год с запятой в указанной позиции.
SYEAR или YEAR	Год, записанный словами, а не цифрами; если указано 'S' то перед датами до нашей эры ставится '-'.
RR	Последние 2 цифры года; для указания года в других столетиях.
BC или AD	BC- до нашей эры(до н.э.); AD – нашей эры
B.C. или A.D.	B.C.- до нашей эры(до н.э.); A.D. – нашей эры
Q	Квартал (1, 2, 3, 4; JAN-MAR=1).
MM	Месяц(01-12; JAN=1).
RM	Нумерация месяцев римскими цифрами(I-XII; JAN=I).
MONTH	Название месяца, дополненное пробелами до 9-ти символов.
MON	Сокращенное название месяца.
WW или W	Неделя года (1-52) или месяца (1-5).
IW	Неделя года (1-52 или 1-53) по стандарту ISO.
DDD или DD или D	День года (1-366) или месяца (1-31) или недели (1-7).
DAY	Название дня, дополненное пробелами до 9-ти символов.
DY	Сокращенное название дня.
J	Дата юлианского календаря; число дней, считая с первого января 4712 года до н.э.
AM или PM	AM –до полудня PM- после полудня
A.M. или P.M.	A.M. –до полудня P.M.- после полудня
HH или HH12	Час дня (1-12).
HH24	Час дня (0-23).
MI	Минута (0-59)
SS или SSSSS	Секунда (0-59) или количество секунд после полуночи (0-86399).
-,,::	Знаки пунктуации.
“...текст...”	Текст воспроизводится в возвращенном значении.



Пример 11:

Приведите пример использования форматных масок

```

1  SELECT to_char(
2     sysdate
3     , 'dd.mm.yyyy hh24:mi:ss'
4  )
5  FROM dual;
```

	TO_CHAR(SYSDATE, 'DD.MM.YYYYHH24:MI:SS')
1	22.04.2025 18:45:20

Префиксы и суффиксы элементов формата даты

К элементам формата даты можно добавлять следующие префиксы:

FM	“Режим заполнения” подавляет заполнение пробелами, когда стоит перед MONTH или DAY
FX	“Точный формат”. Этот модификатор задает точное соответствие символического аргумента и форматной маски даты в функции TO_DATE.

К элементам формата даты можно добавлять следующие суффиксы:

TH	Порядковый номер (“DDTH” для “4TH”).
SP	Номер, записанный словами (“DDSP” для “FOUR”).
SPTH и THSP	Порядковый номер, записанный словами (“DDSPTH” для “FOURTH”).

Прописные и строчные буквы в элементах формата даты.

Следующие строки задают вывод прописными буквами, вывод прописными буквами только начальных букв слов, или вывод строчными буквами.

Прописные	Прописная начальная	Строчные
DAY	Day	.day
DY	Dy	.dy
MONTH	Month	.month
MON	Mon	.mon
YEAR	Year	.year
AM	Am	.am
PM	Pm	.pm
A.M.	A.m.	a.m.
P.M.	P.m.	p.m.

Если к элементу формата даты добавляется префикс или суффикс, то регистр (прописные, строчные буквы) определяется элементом формата, а не префиксом или суффиксом. Например, ‘ddTH’ задает “04th” а не “04TH”.



Пример 12:

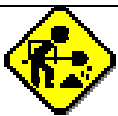
Приведите пример использования префиксов и суффиксов

```
1 ∨ SELECT to_char(  
2     sysdate  
3     , 'dd.mmfx, day - hh24fm:mifm:ssfx'  
4 ∨ )  
5     FROM dual;
```

	TO_CHAR(SYSDATE, 'DD.MMFX, DAY - HH24FM:MIFM:SSFX')
1	22.04, вторник - 18:51:59

Функции преобразования

Функция	Возвращаемое значение
CHARTOROWID(char)	Char преобразуется из типа данных CHAR в тип данных ROWID
CONVERT(char, dest_char_set [,source char set])	Преобразует символьную строку из набора символов source char set в набор символов dest_char_set
HEXTORAW (char)	Преобразует значение char, содержащее шестнадцатичные цифры, в значение типа RAW
RAWTOHEX (raw)	Преобразует raw в символьное значение, содержащее его шестнадцатичный эквивалент
ROWIDTOCHAR (rowid)	Преобразует значение типа ROWID в значение типа CHAR
TO_CHAR (expr [,fmt [,’nls_num_fmt’]])	Преобразует значение expr типа DATE или NUMBER в значение типа CHAR по формату форматной маски fmt. Если fmt отсутствует, значения типа DATE преобразуются по формату, заданному по умолчанию, и значения типа NUMBER- в значение типа CHAR с шириной, достаточной для того, чтобы вместить все значащие цифры. Значение ‘nls_num_fmt’ определяет связанные с языком форматные маски. В Trusted ORACLE преобразует значения MLS или MLS_LABEL в значение типа VARCHAR2
TO_DATE (char[,fmt [,’nls_lang’]])	Преобразует char в значение типа DATE с помощью форматной маски fmt. Если fmt опускается, используется форматная маска для даты, принятая по умолчанию ‘nls_lang’ задает язык, используемый в названиях месяцев и дней
TO_MULTI_BYTE (char)	Преобразует однобайтовые символы, имеющие многобайтовые эквиваленты, в соответствующие многобайтовые символы
TO_NUMBER (char [,fmt [,’nls_lang’]])	Преобразует char, содержащее число в формате, указанном параметром fmt, в значение типа NUMBER. ‘nls_lang’ задает язык, определяющий символы валют и числовые разделители
TO_SINGLE_BYTE (char)	Преобразует многобайтовые символы, имеющие однобайтовые эквиваленты, в соответствующие однобайтовые символы



Пример 12:

Приведите пример использования функций преобразования

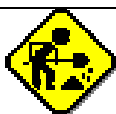
```
SELECT TO_DATE('22.02.2025')
FROM dual;
```

	TO_DATE (' 22.02.2025')
1	22.02.25

Элементы формата числа для TO_CHAR

В следующей таблице перечислены элементы формата числа. Комбинацию этих элементов можно использовать как аргумент *fmt* функции TO_CHAR.

Элемент формата	Пример	Описание
9	'999'	Количество девяток указывает число возвращаемых значащих цифр.
0	'0999'	Добавляет нули перед числом.
\$	'\$9999'	Добавляет знак доллара перед числом.
B	'B9999'	Заменяет нулевые значения пробелами.
MI	'99999MI'	Возвращает знак '-' после отрицательных значений.
S	S9999	Возвращает знак '+' для положительных значений и знак '-' для отрицательных значений в указанную позицию.
PR	'9999PR'	Возвращает отрицательные значения в <угловых скобках>.
D	99D99	Возвращает символ, представляющий десятичную точку, в указанную позицию.
C	9G999	Возвращает символ разделения цифр на группы в указанную позицию.
C	C999	Возвращает международной знак валюты в указанную позицию.
L	L999	Возвращает знак местной валюты в указанную позицию.
,	'9,999'	Возвращает запятую в указанную позицию.
.	'99.99'	Возвращает точку в указанную позицию.
V	'999V99'	Умножает значение на 10^n , где <i>n</i> количество девяток после 'V'.
EEEE	'9.999EEEE' E'	Возвращает значение в нормализованной форме. В <i>fmt</i> должно быть ровно четыре буквы 'E'.
RN или rn	RN	Возвращает римские цифры прописными или строчными буквами (целое число в диапазоне от 1 до 3999).
DATE	'DATE'	Возвращает значение, преобразованное из даты юлианского календаря в формат 'MM/DD/YY'.



Пример 13:

Приведите пример использования форматных масок для to_char()

```
SELECT to_char(
    extract(YEAR FROM current_date)
    , 'FMRM'
) roman_year
FROM dual;
```

	ROMAN_YEAR
1	MMXXV

Другие функции

Функция	Возвращаемое значение
DECODE (expr, search1, return1, [search2, return2,]...[default])	Если expr равно search, возвращается соответствующий результат return. Если совпадающей пары не найдено, возвращается default.
DUMP (expr[, return_format [, art_position[, length]]])	Expr во внутреннем формате Oracle
GREATEST (expr[, expr]...)	Наибольшее значение expr
LEAST (expr[, expr]...)	Наименьшее значение expr
NVL (expr1, expr2)	Возвращает expr2, если expr1 имеет пустое значение, в противном случае возвращает expr1.
UID	Целое число, которое уникально идентифицирует текущего пользователя.
USER	Имя текущего пользователя ORACLE.
USERENV (option)	Возвращает информацию о текущем сеансе. Аргументы помещаются в одиночных кавычках. Аргументы: ENTRYID, SESSIONSID, TERMINAL, LANGUAGE или LABEL.
VSIZE (expr)	Длина в байтах внутреннего представления для expr.



Пример 14:

Приведите пример использования дополнительных функций

```

1  SELECT user
2     | ,userenv('SESSIONID')
3  FROM dual;
```

	USER	USERENV('SESSIONID')
1	SYSTEM	110152

▼ DECLARE

```

months NUMBER;
days   NUMBER;
hours   NUMBER;
now     DATE := :now;
sessia  DATE := :sessia;
```

▼ BEGIN

```

dbms_output.put_line('До начала сессии '
|| to_char(floor(months_between(
    sessia
    ,now
)))
|| ' месяцев или '
|| to_char(floor(sessia - now))
|| ' дней или '
|| to_char(floor((sessia - now) * 24))
|| ' минут');
```

END;

/

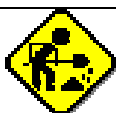
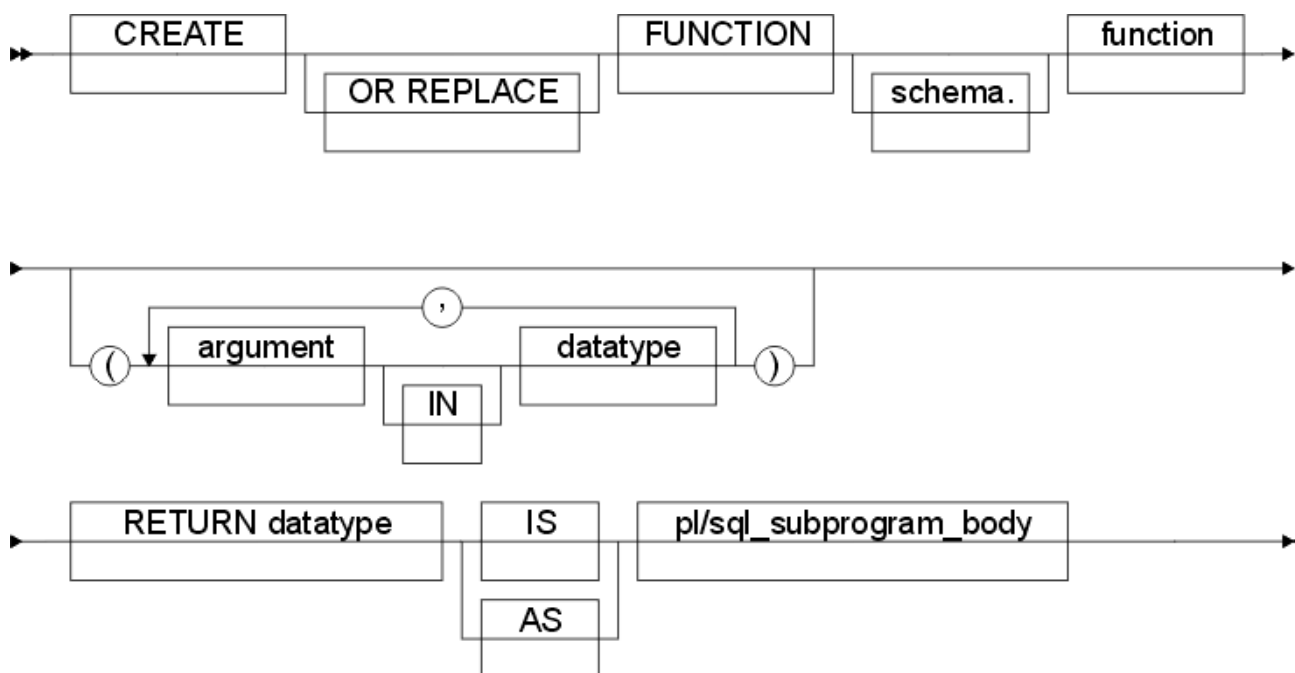


За
Па
вы
сес

До начала сессии 0 месяцев или 24 дней или 576 минут

НАИМЕНОВАННЫЕ PL/SQL БЛОКИ

Создание функции:



Пример 15:

Разработать функцию, вычисляющую площадь круга

```

create or replace
function lokrug(radius
real) return real is
  l real;
begin
  l:=3.14*radius*2;
  return(l);
end;
/

```

```

create or replace
function lokrug1(radius
real) return real is
begin

return(3.14*radius*2);

end;

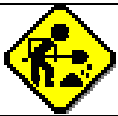
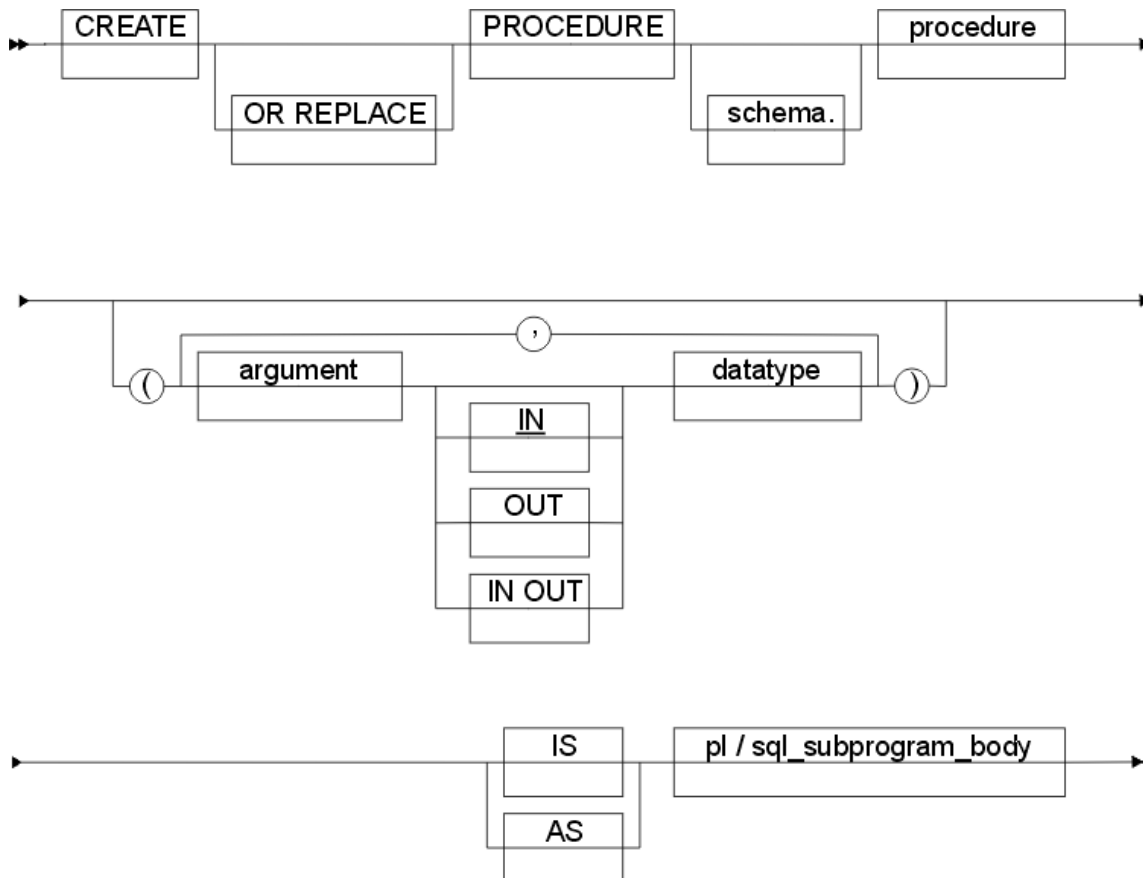
```

```

CREATE OR REPLACE FUNCTION skrug (
  radius REAL
) RETURN REAL IS
  s REAL;
BEGIN
  s := 3.14 * ( radius * * 2 );
  RETURN ( s );
END;

```

Создание процедуры



Пример 16:

Разработать процедуру, вычисляющую площадь круга и длину окружности

```

Prompt формируем отчет по
всем объектам созд. за
промежуток времени
create or replace procedure
prot_user is
db date:=&db;
de date:=&de;
a char(20); b char(20); c
date; e date;
begin
    select
    substr(object_name,1,20)
name,
substr(object_type,1,20)
type_obj,
created, last_ddl_time
into a,b,c,e
from user_objects
where last_ddl_time>=db
and
last_ddl_time<=de;
end;/

```

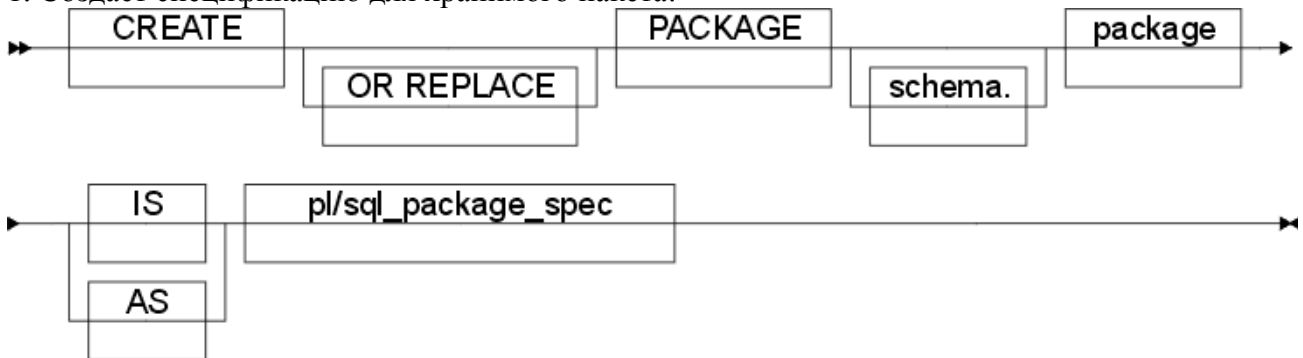
```

CREATE OR REPLACE PROCEDURE krug_params (
    radius IN REAL
    ,s      OUT REAL
    ,l      OUT REAL
) IS
BEGIN
    s := 3.14 * ( radius ** 2 );
    l := 2 * 3.14 * radius;
END:

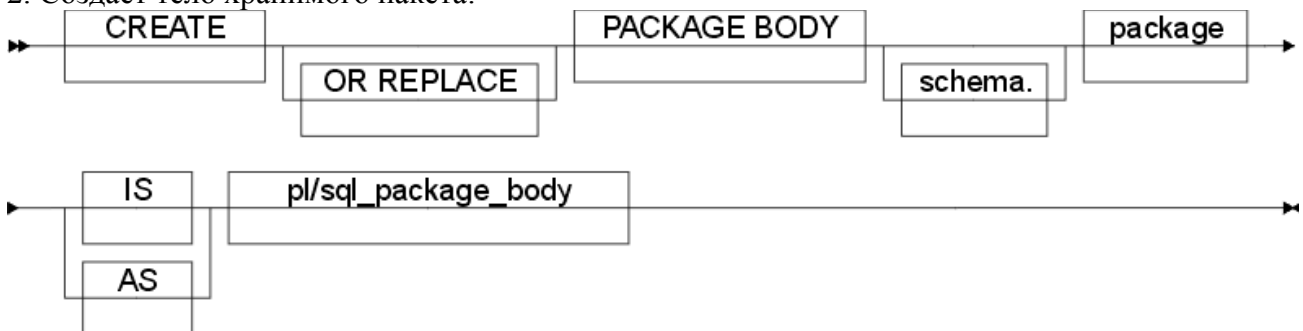
```


Создание пакета

1. Создает спецификацию для хранимого пакета:



2. Создает тело хранимого пакета:



Пример 17:

Разработать пакет – геометрический калькулятор, который по введенному значению радиуса вычисляет площадь круга и длину окружности (используя функции и процедуры)

```
CREATE OR REPLACE PACKAGE calc IS
```

```
    FUNCTION square (
        radius IN REAL
    ) RETURN REAL;
    PROCEDURE length (
        radius IN REAL
        ,len    OUT VARCHAR
    );
END calc;
```

```
/
```

```
CREATE OR REPLACE PACKAGE BODY calc IS
```

```
    FUNCTION square (
        radius IN REAL
    ) RETURN REAL IS
    BEGIN
        RETURN ( 3.14 * radius ** 2 );
    END;
```

```
    PROCEDURE length (
        radius IN REAL
        ,len    OUT VARCHAR
    ) IS
```

```
    BEGIN
        len := 2 * 3.14 * radius;
        dbms_output.enable;
        dbms_output.put_line(len);
    END;
```

```
END calc;
```

```
DECLARE
```

```
    sqr REAL;
    len REAL;
    rad REAL := 1;
```

```
BEGIN
```

```
    sqr := calc.square(rad);
    calc.length(
        rad
        ,len
    );
    dbms_output.put_line('Square = ' || to_char(sqr));
    dbms_output.put_line('Length = ' || to_char(len));
```

```
END;
```

```
/
```

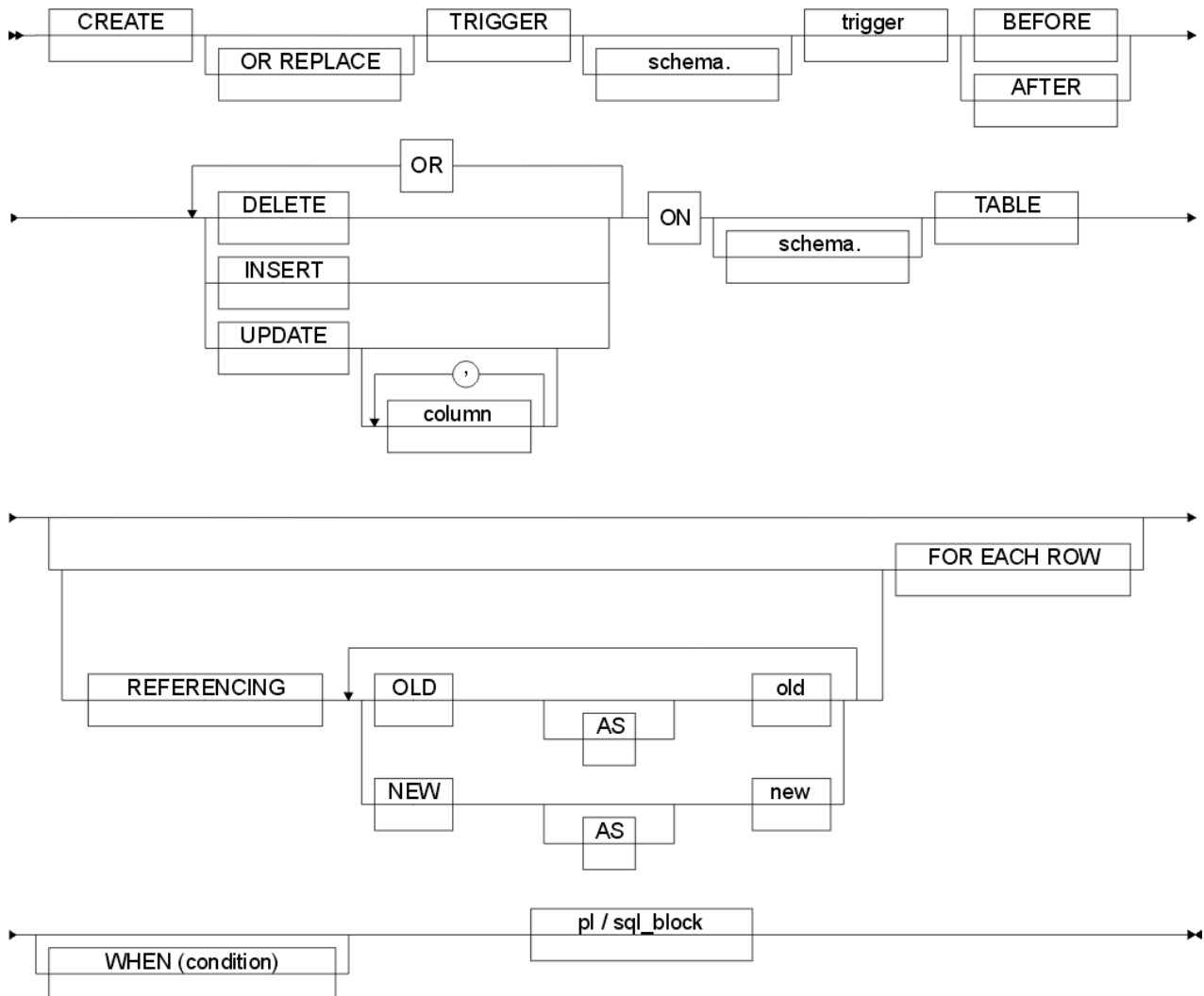
Package Body CALC compiled

6.28

Square = 3.14

Length = 6.28

Создание триггера



Пример 18:

Разработать триггеры для всех таблиц вашей схемы, обеспечивающих автоматическую вставку уникального значения поля ID.

```

CREATE TABLE test1 (
  id  NUMBER
  ,data VARCHAR(255)
);

CREATE SEQUENCE test1_seq;

CREATE OR REPLACE TRIGGER test1_trig BEFORE
  INSERT ON test1
  FOR EACH ROW
BEGIN
  SELECT test1_seq.NEXTVAL
  INTO :new.id
  FROM dual;
END;
    
```



```
SQL> insert into test1 values(0,'111') ;
```

1 row inserted.

```
SQL> select * from test1;
```

ID	DATA
2	111
4	111
6	111
8	111
10	111

Создание констрейтов

	Пример 19: Разработать констрейты для всех связей таблиц вашей инфологической модели (см сем 1), обеспечивающих ссылочную целостность данных.
<pre>ALTER TABLE abonents ADD CONSTRAINT C_abonents_ab_kateg FOREIGN KEY (ab_kateg) REFERENCES list_kategs(lk_id); ALTER TABLE telefons ADD CONSTRAINT C_telefons_tel_ab_num FOREIGN KEY (tel_ab_num) REFERENCES abonents(ab_num);</pre>	<pre>ALTER TABLE defect ADD (CONSTRAINT c_comp_fk FOREIGN KEY (comp_id) REFERENCES components); ALTER TABLE products ADD (CONSTRAINT c_empl_id FOREIGN KEY (prds_empl_id) REFERENCES employees); ALTER TABLE products ADD (CONSTRAINT c_docs_id FOREIGN KEY (prds_docs_id) REFERENCES documentation);</pre>
 <p>prompt Неправильная вставка данных при констрейтах</p> <pre>insert into telefons (tel_num,tel_ab_num) values('12345',1);</pre> <p>prompt Правильная вставка данных при констрейтах</p> <pre>insert into telefons (tel_num,tel_ab_num) values('12345',null); insert into telefons (tel_num,tel_ab_num) values('20012',10005); commit;</pre>	<p>Пример 20: Проверить корректную работу созданных констрейтов</p> <pre> √ INSERT INTO defect (dfct_date ,dfct_type ,dfct_desc ,dfct_name ,dfct_id √) VALUES (TO_DATE('22.04.2025') , 'type' , 'desc' , 'name' , 666); </pre> <p>ORA-02291: нарушено ограничение целостности (c_comp_fk) - исходный ключ не найден https://docs.oracle.com/error-help/db/ora-02291/</p>

Контрольные вопросы

1. Принципы разработки ненаименованных PL/SQL блоков
2. Принципы разработки и использования наименованных PL/SQL блоков.