

Основы Интернет-технологий

ОСНОВЫ PL/SQL

Лекция 10

Язык PL/SQL

Язык PL/SQL является составной частью во многих продуктах Oracle. Сервер Oracle включает поддержку языка PL/SQL, предоставляя пользователю возможность создавать и использовать на сервере процедуры и триггеры базы данных, выполняющие задачи конкретного приложения. Программы, созданные на языке PL/SQL, могут работать совместно в различных частях прикладной системы, построенной на средствах Oracle. Например, в приложении, использующем работу с формами, триггер может вызывать для выполнения некоторого действия хранимую процедуру.

PL/SQL – это процедурный язык в SQL (Procedural Language/SQL). PL/SQL расширяет SQL конструкциями, присутствующими в процедурных языках программирования. Основная единица программы PL/SQL - это блок. Все программы PL/SQL состоят из блоков, которые могут быть вложены друг друга. Обычно, каждый блок соответствует какой-либо логической единице программы.

Операции

Арифметические операторы

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

Операции

Операторы сравнения

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

Операции

:=	Присвоение
(Начало списка или подвыражения
)	Конец списка или подвыражения
,	Отдельные элементы списка (как в списке параметров)
..	Оператор диапазона используется в операторах FOR-IN
 	Конкатенация строк
=>	Ассоциация (используется в списке параметров)
;	Конец выражения
%	Атрибут курсора или типа объекта
.	Спецификация объекта
@	Индикатор удаленной базы данных
'	Начало/конец строки символов
:	Индикатор внешней переменной
&	Индикатор связанной переменной

Структура программы на PL/SQL

Программа на PL/SQL состоит из трех блоков: описаний, исполнительного и блока обработки исключительных ситуаций. Исполнительный блок может быть структурирован с использованием операторных скобок BEGIN и END.

Синтаксически программа на PL/SQL оформляется следующим образом:

```
[DECLARE]
    -- объявления
BEGIN
    -- операторы
[EXCEPTION]
    -- исключения
END;
```

В блоке DECLARE описываются переменные, константы и определяемые пользователем типы данных:

Identifier [CONSTANT] datatype [NOT NULL] [:=DEFAULT | expr];

CONSTANT — это ограничение на переменную, которая не меняется в программе, но должна быть инициализирована. Две переменные в разных блоках могут иметь одинаковые имена, переменным внутри одного блока не рекомендуется давать те же имена, что и имена столбцов в таблицах в этом блоке.

Структура программы на PL/SQL

В PL/SQL существует блоки трех типов:
безымянный блок,
процедурный блок (PROCEDURE),
функция (FUNCTION).

1. Безымянный блок

```
[DECLARE]
    -- объявления
BEGIN
    -- операторы
[EXCEPTION]
    -- исключения
END;
```

Структура програми на PL/SQL

2. PROCEDURE

```
PROCEDURE name IS  
BEGIN  
    -- операторы  
[EXCEPTION]  
    -- исключения  
END;
```

3. FUNCTION

```
FUNCTION name RETURN datatype IS  
BEGIN  
    -- операторы  
[EXCEPTION]  
    -- исключения  
END;
```


Структура программы на PL/SQL

Команды установки переменных окружения `SERVEROUTPUT` и `ECHO` определяют режим вывода на терминал пользователя.

Системная процедура `DBMS_OUTPUT.PUT_LINE` обеспечивает вывод данных на терминал пользователя, а функция `EXP` вычисляет экспоненту. Символ `/` указывает на завершение текста процедуры и является командой к интерпретации и выполнению процедуры

```
SQL>set serveroutput on;
```

```
SQL>set echo on;
```

```
SQL>DECLARE
```

```
2  Header1 CONSTANT VARCHAR2(20):='кспонента двух равна';
```

```
3  Header2 CONSTANT VARCHAR2(20):='Экспонента трех равна';
```

```
4  Arg NUMBER:=2;      — здесь задается начальное значение аргумента
```

```
6  BEGIN
```

```
7  DBMS_OUTPUT.PUT_LINE(Header1 || Exp(Arg));
```

```
8  Arg:=Arg+1;
```

```
9  DBMS_OUTPUT.PUT_LINE(Header2 || Exp(Arg));
```

```
10 END;
```

```
11 /
```

Переменные и типы

Программа PL/SQL и база данных обменивается информацией через переменные. Каждая переменная должна иметь тип. В качестве типа переменной может быть:

- Один из типов, который используется в SQL для определения типа столбца.
- Обобщенный тип, используемый в PL/SQL, подобный типу NUMBER.
- Объявленный (пользовательский), но такой, какой имеет один из столбцов.

Наиболее часто используемый обобщенный тип – это NUMBER. В переменных типа NUMBER могут храниться как целые, так и вещественные числа. Наиболее обычно используемый символьный строковый тип - VARCHAR (n), где n - максимальная длина строки в байтах. Длина указывается обязательно, значения по умолчанию нет.

Например:

```
DECLARE  
    price NUMBER;  
    myBeer VARCHAR(20);
```

Переменные и типы

PL/SQL поддерживает тип BOOLEAN, хотя Oracle не допускает его в качестве типа столбца.

Типы в PL/SQL могут быть “базируемыми”. Во многих случаях, PL/SQL переменная будет использоваться, чтобы работать с данными, хранимыми в таблицах. В этом случае, необходимо, чтобы переменная имела тот же самый тип, что и столбец таблицы. Если имеется несоответствие типов, команды присваивания переменных и сравнения переменных могут выполняться не так, как ожидается. Чтобы избежать этого, вместо жесткого указания типа переменной, можно использовать оператор **TYPE %**.

Например:

```
DECLARE
```

```
myBeer Beers.name%TYPE;
```

После выполнения этой команды переменная myBeer будет иметь тот же тип, что и столбец name таблицы Beers.

Переменные и типы

Кроме этого, можно определить переменную-запись, составляющие которой имеют тип и длину, как и несколько столбцов таблицы. Самый простой способ определить такую переменную – это применить оператор %ROWTYPE к имени таблицы.

Например:

```
DECLARE  
    beerTuple Beers%ROWTYPE;
```

После выполнения этой команды переменная beerTuple будет переменной-записью с полями name и manufacture, такими же, как и в структуре таблицы Beers(name, manufacture).

Переменные и типы

Начальное значение любой переменной, независимо от типа, - NULL. Для присваивания значений переменным используется оператор ": = ". Присваивание значения может происходить немедленно после того, как объявлен тип переменной или где-нибудь в разделе выполняемого кода программы. Например:

```
DECLARE
  a NUMBER := 3;
BEGIN
  a := a + 1;
END;
```

/

Оператор ветвления IF

В PL/SQL предусмотрено несколько операторов, с помощью которых можно управлять выполнением программы.

Oracle использует следующий синтаксис конструкции ветвления в PL/SQL:

```
IF условие1 THEN
    операторы1;           — ветвь 1
[ELSIF условиеK THEN
    операторыK;]         — ветвьK
[ELSE
    операторыN;]         — ветвь альтернативы
— операторы альтернативы
END IF;
```

Оператор ветвления CASE

При большом количестве ветвлений и/или необходимости присваивания значений удобно применить оператор CASE:

Простой оператор CASE:

```
CASE выражение  
WHEN результат1 THEN значение1  
WHEN результатK THEN значениеK  
[ELSE] значениеN  
END CASE;
```

Поисковый оператор CASE:

```
CASE  
WHEN выражение1 THEN значение1  
WHEN выражение2 THEN значение2  
  
...  
WHEN выражениеN THEN значениеK  
[ELSE] значениеN  
END CASE;
```

Оператор ветвления CASE

Выражения CASE – используются внутри выражений когда нужно присвоить значение

Синтаксис

```
Простое_выражение_CASE:= CASE выражение  
WHEN результат1 THEN значение1
```

...

```
WHEN результатK THEN значениеK  
[ELSE] значениеN  
END;
```

```
Поисковое_выражение_CASE:= CASE  
WHEN выражение1 THEN значение1  
WHEN выражение2 THEN значение2
```

...

```
WHEN выражениеN THEN значениеK  
[ELSE] значениеN  
END;
```


Оператор цикла

В PL/SQL есть три типа оператора цикла: **LOOP, FOR LOOP, WHILE LOOP**:

LOOP

Операторы

EXIT [WHEN условие] ;

END LOOP;

Последовательность операторов между LOOP и END LOOP многократно повторяется до тех пор, пока условие не примет значение TRUE.

Условие может принимать одно из трех значений: TRUE, FALSE, NULL.

Этот оператор выполняется, по крайней мере, хотя бы один раз, если будет отсутствовать EXIT, то последовательность операторов будет выполняться бесконечно. Если условие принимает значение TRUE, то будет выполняться следующий за LOOP оператор

Оператор цикла FOR

```
FOR counter IN [REVERSE] lower_bound .. upper_bound LOOP
```

```
  Оператор1.;
```

```
  ОператорК;
```

```
END LOOP;
```

REVERSE означает изменение параметра цикла от верхней границы до нижней границы, при этом нижняя граница в операторе цикла опять стоит первой. Объявлять параметр цикла counter в операторе DECLARE нет необходимости, он по умолчанию объявляется как целый, lower_bound — нижняя граница изменения параметра цикла, upper_bound верхняя граница изменения параметра цикла

Оператор цикла WHILE

WHILE условие LOOP

Оператор1;

ОператорК;

END LOOP;

Последовательность операторов между LOOP и END LOOP повторяется до тех пор, пока условие имеет значение TRUE. Условие проверяется до выполнения последовательности операторов, если условие принимает значение FALSE, то последовательность операторов перестает выполняться. Этот оператор может ни разу не проработать, если условие перед выполнением оператора имеет значение FALSE.

Метки

Метки следует использовать для структурирования программы и более гибкого управления выходом из вложенных циклов.

Синтаксис метки: <<имя_метки>>

Например

```
<<All_emp>>
```

```
FOR emp_rec IN emp_cur LOOP
```

```
.....
```

```
END LOOP All_emp;
```

Метки

Метки следует использовать для структурирования программы и более гибкого управления выходом из вложенных циклов.

Синтаксис метки: <<имя_метки>>

Например

```
<<All_emp>>
```

```
FOR emp_rec IN emp_cur LOOP
```

```
.....
```

```
END LOOP All_emp;
```