

Границы моего языка означают границы
моего мира.

Людвиг Виттгенштейн

ОСНОВЫ ЯЗЫКА PL/SQL

Общие сведения

Алфавит

Алфавит:

- Латинские буквы: a ... Z A ... Z
 - Арабские цифры: 0 ... 9
 - Символы табуляция, пробел и возврат каретки ("пропуски")
 - Символы ()+*/<>=!~::~'@%,"#\$^&_[{ }?[]
- # \$ _ – дополнительные символы, которые можно использовать как буквы

PL/SQL не различает прописных и строчных букв, и рассматривает строчные буквы как эквиваленты соответствующих прописных букв, исключая строковые и символьные литералы.

Лексические единицы

Строка текста программы PL/SQL распадается на группы символов, называемые ЛЕКСИЧЕСКИМИ ЕДИНИЦАМИ, которые можно классифицировать следующим образом:

- разделители (простые и составные символы),
- идентификаторы, в том числе зарезервированные слова,
- литералы,
- комментарии.

Простые символы

Простые символы кодируются как одиночные символы:

- + оператор сложения
- оператор вычитания/отрицания
- * оператор умножения
- / оператор деления
- = оператор сравнения
- < оператор сравнения
- > оператор сравнения
- (ограничитель выражения или списка
-) ограничитель выражения или списка
- ; терминатор предложения
- % индикатор атрибута
- , разделитель элементов
- . селектор компоненты
- @ индикатор удаленного доступа
- ' ограничитель символьной строки
- " ограничитель идентификатора
- : индикатор хост-переменной

Составные символы

Составные символы кодируются как пары символов:

- ** оператор возведения в степень
- <> оператор сравнения
- != оператор сравнения
- ~= оператор сравнения
- ^= оператор сравнения
- <= оператор сравнения
- >= оператор сравнения
- := оператор присваивания
- => оператор ассоциации
- .. оператор интервала
- || оператор конкатенации
- << ограничитель метки
- >> ограничитель метки
- индикатор однострочного комментария
- /* (начальный) ограничитель многострочного комментария
- */ (конечный) ограничитель многострочного комментария.

Идентификаторы

Идентификатор – последовательность символов, которая начинается с латинской буквы (или #, \$, _) и содержит буквы, цифры или заменяющие буквы символы.

Длина идентификатора не может превышать 30 символов.

Для большей гибкости, PL/SQL позволяет заключать идентификаторы в двойные кавычки. Такой идентификатор может содержать любую последовательность печатных символов, включая пробелы, но исключая двойные кавычки. Например:

"X+Y"

"last name"

"on/off switch"

"employee(s)"

*** header info ***"

Литералы

ЛИТЕРАЛ – это явное число, символ, строка или булевское значение, не представленное идентификатором. Примерами могут служить числовой литерал 147 и булевский литерал FALSE.

Числовые литералы

Целочисленный литерал – это целое число с необязательным знаком и без десятичной точки. Примеры:

0 30 6 -14 0 +32767

Вещественный литерал – это целое или дробное число с необязательным знаком и с десятичной точкой. Примеры:

6.6667 0.0 -12.0 +8300.00 .5 25. 1.0E-7 3.14159e0

Булевские литералы

Булевские литералы – это predetermined значения TRUE и FALSE, а также "не-значение" NULL, которое обозначает отсутствие, неизвестность или неприменимость значения.

Булевские литералы НЕ являются строками.

Литералы

Символьные литералы

Символьный литерал – это одиночный символ, окруженный одиночными апострофами. Примеры:

```
'Z' '%' '7' ' ' 'z' '('
```

Строковые литералы

Строковый литерал – это последовательностью из нуля или более символов, заключенной в апострофы. Примеры:

```
'Hello, world!' 'XYZ Corporation' '10-NOV-91' '$1,000,000'
```

Все строковые литералы, за исключением пустой строки ("), имеют тип CHAR. Если необходимо включить апостроф в литерал, его необходимо изображать в виде двойного апострофа ("), что не то же самое, что двойная кавычка ("):

```
'Don"t leave without saving your work.'
```

Комментарии

-- однострочный комментарий

```
/* многострочный комментарий */
```

Комментарии нельзя вкладывать друг в друга.



Семейства типов данных PL/SQL



Типы данных PL/SQL



Типы данных PL/SQL



Типы данных PL/SQL



Типы данных PL/SQL



Преобразования типов данных

Неявные преобразования

Явные преобразования

Замечания по типам данных PL/SQL

Тип NUMBER: отрицательный масштаб вызывает округление слева от десятичной точки. Например:

NUMBER(5, -2): 12345 будет храниться как 12300

Идентификатор строки с помощью функции ROWIDTOCHAR преобразуется из двоичного значения в 18-байтовую символьную строку, возвращая ее в формате

BBBBBBBBB.RRRR.FFFF

где

BBBBBBBBB – номер блока в файле базы данных (блоки нумеруются с 0),

RRRR – номер строки в блоке (строки нумеруются с 0),

FFFF – номер файла базы данных.

Все эти числа шестнадцатеричные. Например, идентификатор строки 0000000E.000A.0007 указывает на 11-ю строку 15-го блока в 7-м файле базы данных.

Структура программы

PL/SQL – это язык, структурированный блоками.

Объявления и инициализация констант и переменных:

<идентификатор> [CONSTANT] <тип> [NOT NULL] [:=<нач. значение>];

part_no NUMBER(4);

in_stock BOOLEAN NOT NULL := FALSE;

minimum_balance CONSTANT REAL DEFAULT 10.00;

pi CONSTANT REAL := 3.14159;

По умолчанию переменные инициализируются значением NULL.

Атрибуты

Переменные и константы PL/SQL имеют АТТРИБУТЫ, т.е. свойства, позволяющие ссылаться на тип данных и структуру объекта, не повторяя его объявление.

Аналогичные атрибуты имеются у таблиц и столбцов базы данных, что позволяет упростить объявления переменных и констант.

Атрибут %TYPE

Представляет тип данных переменной, константы или столбца. Примеры:

```
str1    varchar2(100);  
str2    str1%TYPE;  
ename   emp.name%TYPE;
```

Атрибут %ROWTYPE

В PL/SQL для группирования данных используются записи. Запись состоит из нескольких полей, в которых могут храниться значения данных. Атрибут %ROWTYPE обозначает тип записи, представляющей строку в таблице. Примеры объявления:

```
r_emp    emp%ROWTYPE;  
r_depart depart%ROWTYPE;
```




Список ключевых слов PL/SQL

Сфера и видимость

Ссылки на идентификатор разрешаются согласно его сфере и видимости. Идентификаторы, объявленные в блоке PL/SQL, считаются локальными в этом блоке и глобальными для всех его подблоков.

Преимущество имен

Имена столбцов



Имена переменных и констант



Имена таблиц

Старшинство
операций

Обработка пустых значений

Сравнения, в которых участвует NULL, всегда дают NULL;
применение NOT к значению NULL дает NULL;
в предложениях условного управления, если условие дает NULL,
соответствующая группа предложений не выполняется.
PL/SQL трактует любую строку нулевой длины как NULL.

Примеры:

```
x := 5;
```

```
y := NULL;
```

```
IF x != y THEN -- это условие даст NULL, а не TRUE
```

```
-- ряд предложений; -- не выполняется
```

```
END IF;
```

Обработка пустых значений

Если функции передается пустой аргумент, она возвращает NULL, за исключением следующих трех случаев:

DECODE

Функция DECODE сравнивает свой первый аргумент с одним или несколькими поисковыми выражениями, которые спарены с результирующими выражениями.

```
credit_limit := DECODE(rating, NULL, 1000, 'B', 2000, 'A', 4000);
```

NVL

Если ее первый аргумент есть NULL, функция NVL возвращает значение своего второго аргумента:

```
start_date := NVL(hire_date, SYSDATE);
```

REPLACE

Если ее второй аргумент NULL, функция REPLACE возвращает значение своего первого аргумента, независимо от того, присутствует ли необязательный третий аргумент

```
new_string := REPLACE(old_string, NULL, my_string);
```

Если ее третий аргумент NULL, функция REPLACE возвращает значение своего первого аргумента, из которого удалены все вхождения второго аргумента.

Сравнение строк и числовых значений

При сравнении значений типа CHAR более короткая строка дополняется пробелами до более длинной и сравнение происходит корректно.

При сравнении значений CHAR с VARCHAR дополнение пробелами не происходит, поэтому сравнение может быть некорректным:

a CHAR(10) := 'Ann';

b VARCHAR(10) := 'Ann';

IF a=b THEN – эти операции не выполнятся
END IF;

При сравнении целых и действительных чисел результат негарантирован:

count := 1;

IF count = 1.0 THEN ...