**Оглавление**

[Структура PL/SQL блока](#_gjdgxs)

[Раздел объявлений](#_30j0zll)

[Раздел выполнения](#_1fob9te)

[Раздел обработки исключительных ситуаций](#_3znysh7)

[Анонимные блоки](#_2et92p0)

[Синтаксис языка](#_tyjcwt)

[Идентификаторы](#_3dy6vkm)

[Арифметические операторы:](#_1t3h5sf)

[Операторы отношения (используются в логических выражениях):](#_4d34og8)

[Комментарии и метки](#_2s8eyo1)

[Основные типы и структуры данных](#_17dp8vu)

[Типы LOB](#_44sinio)

[Составные типы](#_2jxsxqh)

[Динамические типы](#_z337ya)

[Пользовательские подтипы](#_3j2qqm3)

[Объявление переменных и постоянных](#_1y810tw)

[Функции для работы с NULL](#_4i7ojhp)

[DECODE](#_2xcytpi)

[NVL](#_1ci93xb)

[NVL2](#_3whwml4)

[COALESCE](#_2bn6wsx)

[Неявное преобразование типов](#_qsh70q)

[Явное преобразование типов](#_3as4poj)

[Основные управляющие структуры](#_1pxezwc)

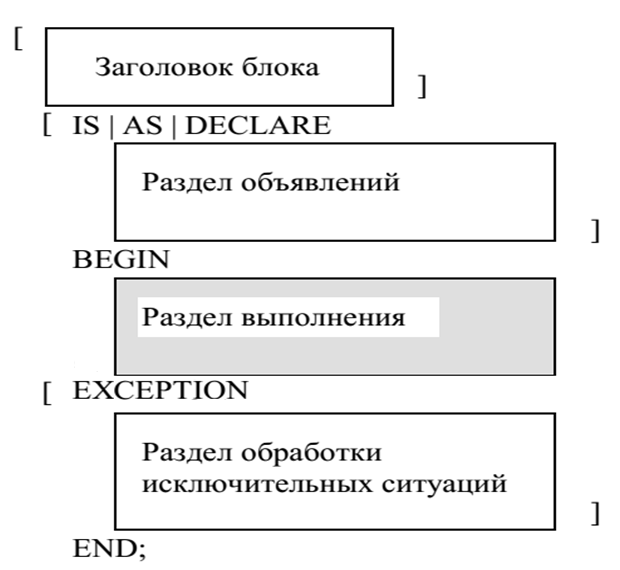
[Выбор по условию](#_49x2ik5)

[Циклы](#_2p2csry)

[Goto (безусловный переход)](#_147n2zr)

[Метки в циклах и блоках](#_3o7alnk)

## Структура PL/SQL блока



Все разделы, кроме раздела выполнения, имеют право отсутствовать.

Раздел выполнения может наряду с программным кодом содержать произвольное число других (вложенных) блоков PL/SQL.

### Раздел объявлений

Содержит объявления переменных, констант, курсоров, исключений, функций и процедур PL/SQL, которые будут использоваться в выполняемой секции и секции исключений

Все, что находится в секции объявлений, принадлежит блоку и может использоваться только внутри него.

### Раздел выполнения

Выполняемая секция начинается с ключевого слова BEGIN и заканчивается либо ключевым словом EXCEPTION, если присутствует раздел исключений, либо ключевым словом END.

### Раздел обработки исключительных ситуаций

Раздел исключений начинается с ключевого слова EXCEPTION и продолжается до конца блока. Каждому исключению соответствует оператор WHEN имя\_исключения, указывающий, что должно быть сделано при возникновении данного исключения.

Все операторы, находящиеся между оператором, вызвавшим ошибку, и секцией исключений, игнорируются.

## Анонимные блоки

Анонимный блок (anonumous block) — это блок PL/SQL без секции заголовка

## Синтаксис языка

### Идентификаторы

К идентификаторам PL/SQL предъявляются следующие требования:

* Иметь не более 30 символов в длину и не содержать пробельных символов (собственно пробелов и знаков табуляции).
* Состоять только из букв, цифр от 0 до 9, символа подчеркивания (\_), знака доллара ($) и знака фунта (#).
* Начинаться с буквы.
* Не совпадать с зарезервированными словами PL/SQL или SQL, которые имеют специальное значение. Например, именем переменной не может быть слово BEGIN или INSERT. Все зарезервированные слова можно увидеть в представлении v$reserved\_words.

|  |  |
| --- | --- |
| := | 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 |

### Арифметические операторы:

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

### Операторы отношения (используются в логических выражениях):

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

### Комментарии и метки

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

### 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 |

#### Null

Это специальное значение – означает отсутствие данных, констатацию того факта, что значение неизвестно. По умолчанию это значение могут принимать переменные всех типов данных, если явно не указанно ограничение NOT NULL, если ограничение указанно тогда надо определять значение по умолчанию. Основная особенность заключается в том, что Null ни равен ничему, даже другому Null.

Для сравнения существует специальное сравнение «is», с помощью которого мы можем сравнивать разные переменные.

Таким образом, Oracle оперирует не двухзначной, а трехзначной логикой

#### Скалярные типы

##### Числовые типы

|  |  |
| --- | --- |
| Data Type | Data Description |
| PLS\_INTEGER or BINARY\_INTEGER (с 10й версии 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 ±2-149 до ±2127∙(2-2-23)) |
| BINARY\_DOUBLE | Числа двойной точности, соответствует формату IEEE 754- формат с плавающей запятой (64 бит, диапозон от ±2-1074 до ±21023∙(2-2-52)) |
| NUMBER [ (p [, s]) ] | С фиксированной или плавающей запятой с абсолютным значением в диапазоне от 1E-130 до (но не включая) 1.0E126. Может содержать ноль.  Precision – общее число значащих цифр (max – 38)  Scale – количество цифр справа от запятой (от -84 до 127) |

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 значние -1, 0, или 1 (полезно при программировании tri-state логики (три состояния)) |
| SIMPLE\_INTEGER | PLS\_INTEGER значение с NOT NULL ограничением |

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 |

##### Строковые типы

|  |  |
| --- | --- |
| Data Type | Data Description |
| CHAR [(size [BYTE | CHAR])] | Строки фиксированной длины до 32767 байт (в Oracle SQL предел в 2000 байт) |
| VARCHAR2(size [BYTE | CHAR]) | Строки переменной длины до 32767 байт (в Oracle SQL предел в 4000 байт) |
| NCHAR[(size)] и NVARCHAR2(size) | NLS-символьные типы. Позволяют обрабатывать символьные данные в  мультибайтовой кодировке Unicode |
| RAW(size) | Байтовая строка переменной длины до 32767 байт (в Oracle SQL2000 байт) |
| LONG | Строки переменной длины до 32767 байт. Тип сохранен для обратной совместимости; в частности встречается в некоторых справочных таблицах |
| LONG RAW | Байтовая строка переменной длины до 32767 байт. Тип сохранен для обратной совместимости версий Oracle |

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

|  |  |
| --- | --- |
| Data Type | Data Description |
| ROWID | Двоичный массив фиксированной длины для хранения физического адреса данных Oracle в шестнадцатеричном в формате OOOOOOFFFBBBBBBRRR |
| 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) |

В Oracle строковая переменная равная '' (пустая строка) эквивалентна null.

##### Типы для моментов и интервалов времени

|  |  |
| --- | --- |
| 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. |
| NTERVAL DAY [(day\_precision)] TO SECOND [(fractional\_seconds\_precision)] | Хранит период времени в днях и секундах.  - day\_precision – это количество 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. |

Поддерживаемые значения:

|  |  |  |
| --- | --- | --- |
| Поле | Диапазоны значения для Datetime | Диапозоны для интревальных значений |
| YEAR | -4712 to 9999 (исключая 0) | Любое ненулевое целое |
| MONTH | 01 до 12 | 0 до 11 |
| DAY | 01 до 31 (ограничено значениями MONTH и YEAR, согласно локальным правилам календаря) | Любое ненулевое целое |
| HOUR | 00 до 23 | 0 до 23 |
| MINUTE | 00 до 59 | 0 до 59 |
| SECOND | 00 до 59.9(n), где 9(n) это установка fractional\_seconds\_precision | 0 до 59.9(n), где 9(n) это установка fractional\_seconds\_precision |
| TIMEZONE\_HOUR | -12 до 14 (учитывает переход на летнее время) | Не используется |
| TIMEZONE\_MINUTE | 00 до 59 | Не используется |
| TIMEZONE\_REGION | Выбирается в представлении V$TIMEZONE\_NAMES | Не используется |
| TIMEZONE\_ABBR | Выбирается в представлении V$TIMEZONE\_NAMES | Не используется |

Операции над типом и результат

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

##### Булевы типы

Фактически – тип для трехзначных переменных с допустимыми значениями TRUE, FALSE и NULL.

В выражениях допустимы только булевы операнды.

### Типы LOB

Large OBjects «большие неструктурированные объекты»

|  |  |
| --- | --- |
| 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) |

### Составные типы

Записи являются примером данных составного типа. В PL/SQL они бывают трех видов:

* воспроизводящими структуру таблицы в БД
* воспроизводящими структуру курсора в программе
* заданными пользователем произвольно

Помимо этого к числу составных типов (допускающих структуру значения) Oracle относит типы объектов и коллекций.

Записи в PL/SQL могут объявляться в разделе объявлений блока или в разделе глобальных описаний пакета.

**CREATE TYPE имя\_типа IS RECORD (объявление переменных через “,” );**

Записи, повторяющие структуру таблицы или курсора, объявляются с помощью атрибута %ROWTYPE.

Записи, задаваемые пользователем, объявляются через предложение TYPE. Из примера видно, что записи могут быть вложенными.

### Динамические типы

|  |  |
| --- | --- |
| %TYPE | Ссылка на тип поля в столбце (v\_id customer.id%type;) |
| %ROWTYPE | Ссылка на структуру таблицы, создается запись (Record) (v\_rec\_customer%rowtype |

### Пользовательские подтипы

**SUBTYPE имя\_подтипа IS базовый\_тип [(ограничения)] [NOT NULL];**

Где

* базовый\_тип – любой скалярный или пользовательский PL/SQL тип данных.
* Ограничения – здесь можно указать точность, размер, масштаб и т.д

## Объявление переменных и постоянных

Общий формат объявления выглядит так:

**имя\_переменной [CONSTANT] тип\_данных [NOT NULL] [{:= | DEFAULT} выражение] ;**

CONSTANT – переменная будет константой.

DEFAULT(:=) задает значение по умолчанию (можно ссылаться на вышеописанные переменные, с уже установленным значением по умолчанию).

## Функции для работы с NULL

### DECODE

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

Description of decode.gif follows

### NVL

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

Description of nvl.gif follows

### NVL2

В случае если expr1 is null, тогда вернет expr2 иначе expr3

Description of nvl2.gif follows

### COALESCE

Возвращает первое не null значение

Description of coalesce.gif follows

## Неявное преобразование типов

Почему нельзя использовать:

* 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 возвратить 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.

Таблица неявных преобразований в выражениях

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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. |

К примеру при сравнении NUMBER и CHAR неявному преобразованию подлежит правый операнд, т.е. CHAR

## Явное преобразование типов

Функции явного преобразования 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

### Выбор по условию

#### Предложение IF-THEN

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;

#### Простой CASE

CASE выражение

WHEN выражение\_для\_сравнения1 THEN программный код1

[WHEN выражение\_для\_сравненияi THEN программный кодi]

…

[ELSE программный код]

END CASE;

#### CASE с поиском

CASE

WHEN условное\_выражение1 THEN программный код1

[WHEN условное\_выражениеi THEN программный кодi]

…

[ELSE программный код]

END CASE;

### Циклы

[ 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.

#### While

WHILE условное\_выражение

LOOP

программный код

END LOOP [слово-комментарий ]

#### For по счетчику

FOR индекс\_цикла IN [REVERSE] нижнее\_значение .. верхнее\_значение

LOOP

программный код

END LOOP [ слово-комментарий ]

индекс\_цикла заводится в PL/SQL автоматически как переменная типа PLS\_INTEGER, и объявлять его не требуется.

указание REVERSE заставить перевернет счетчик и он будет декрементировать от верхнее\_значение до нижнее\_значение

#### For по курсору

FOR индексная\_запись IN [имя\_курсора | явное\_предложение\_SELECT]

LOOP

программный код

END LOOP [ слово-комментарий ];

Объявление индексной\_записи в PL/SQL выполняется автоматически и с типом имя\_курсора%TYPE и самостоятельного объявления не требует.

### Goto (безусловный переход)

GOTO имя\_метки выражение;

Ограничения области действия GOTO:

* нельзя передавать управление внутрь предложения IF, LOOP и вложенного блока
* нельзя передавать управление из одного раздела предложения IF в другой
* нельзя передавать управление извне/внутрь подпрограммы
* нельзя передавать управление из раздела обработки исключительных состояний в основной раздел блока PL/SQL
* нельзя передавать управление из основного раздела блока PL/SQL в раздел обработки исключительных состояний (это можно делать только с помощью RAISE)

Но передавать управление изнутри предложения IF, LOOP и вложенного блока возможно

## Метки в циклах и блоках

