**Вычисления дат с помощью SQL**

**Однострочные функции и операции**

**(MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY и LAST\_DAY)**

* Функция **MONTHS\_BETWEEN** возвращает количество месяцев между двумя заданными датами. Синтаксис:

**MONTHS\_BETWEEN (первая дата, вторая дата)**

Примеры:

SELECT months\_between ('20-SEP-2020', '20-JAN-2020') FROM dual;

SELECT months\_between ('20-SEP-2020', '04-JAN-2020') FROM dual;

* Функция **ADD\_MONTHS** возвращает дату, рассчитанную путем добавления (**Как??)** указанного числа месяцев (целой части нецелого числа месяцев) к значению даты .

Синтаксис функции:

**ADD\_MONTHS (дата, количество месяцев)**

Примеры:

SELECT add\_months('04-JAN-2020', 1) FROM dual;

SELECT add\_months('31-MAY-2020', 2.7) FROM dual;

SELECT add\_months('04-JAN-2020', -12) FROM dual;

* Функция **NEXT\_DAY** возвращает дату первого следующего дня недели.

Синтаксис функции:

**NEXT\_DAY (дата, день недели)**

Пример:.

SELECT next\_day('15-JAN-2020', 'Monday') FROM dual;

* Функция **LAST\_DAY** возвращает дату последнего дня месяца для указанной даты.

Синтаксис функции:

**LAST\_DAY (дата)**

Пример:

SELECT last\_day('01-JAN-2020') FROM dual;

| Unit | Valid format parameters |
| --- | --- |
| Year | SYYYY, YYYY, YEAR, SYEAR, YYY, YY, Y |
| ISO Year | IYYY, IY, I |
| Quarter | Q |
| Month | MONTH, MON, MM, RM |
| Week | WW |
| IW | IW |
| W | W |
| Day | DDD, DD, J |
| Start day of the week | DAY, DY, D |
| Hour | HH, HH12, HH24 |
| Minute | MI |

* Функция **TRUNC**

Синтаксис функции:

TRUNC ( date [, format ] )

Пример:

SELECT

TO\_CHAR(

TRUNC(TO\_DATE( '04-Aug-2017 15:35:32 ', 'DD-Mon-YYYY HH24:MI:SS' )),

'DD-Mon-YYYY HH24:MI:SS'

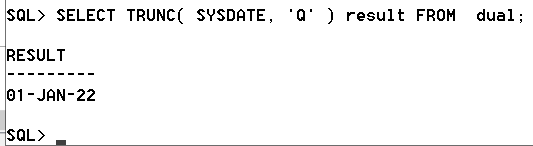
) result

FROM

dual;

SELECT TRUNC(SYSDATE, 'MM' ) result FROM dual;

SELECT TRUNC( SYSDATE, 'Q' ) result FROM dual;



* **Функция TO\_DATE**

**TO\_DATE — преобразует строку в переменную типа DATE.**

**Синтаксис:**

TO\_DATE(строка, формат, NLS\_LANGUAGE);

--Последний параметр NLS\_LANGUAGE задаёт кодировку. Используется редко.

Примеры:

select TO\_DATE('16/10/2016', 'dd/mm/yyyy') from dual;

select to\_date('24.05.1980 14:16:34','dd.mm.yyyy hh24:mi:ss') from dual;

**Примеры**

**Текущая неделя**

-- Сегодня

select trunc (SYSDATE) from dual;

-- Вчера

select trunc (SYSDATE-1) from dual;

-- Завтра

select trunc (SYSDATE+1) from dual;

-- Первый день недели

select trunc(SYSDATE, 'DAY') from dual;

-- Последний день недели

select trunc(SYSDATE, 'DAY')+6 from dual;

**Прошедшая неделя**

-- Первый день прошедшей недели

select trunc(SYSDATE, 'DAY') -7 from dual;

-- **Неделя начинается с воскресенья**

-- Последний день прошедшей недели

select trunc(SYSDATE, 'DAY')-1 from dual;

**Следующая неделя**

-- Первый день следующей недели

select trunc(SYSDATE, 'DAY') +7 from dual;

-- Последний день следующей недели

select trunc(SYSDATE, 'DAY') +13 from dual;

**Текущий месяц**

-- Первый день месяца

select trunc (SYSDATE, 'MM') from dual;

-- Последний день месяца

select trunc (last\_day(sysdate)) from dual;

**Прошедший месяц**

-- Первый день прошлого месяца

select trunc(ADD\_MONTHS(SYSDATE, -1), 'MM') from dual;

-- Последний день прошлого месяца

select trunc (SYSDATE, 'MM') -1 from dual;

**Следующий месяц**

-- Первый день следующего месяца

select trunc (last\_day(sysdate)) +1 from dual;

-- Последний день следующего месяца

select trunc(LAST\_DAY(ADD\_MONTHS(SYSDATE, 1))) from dual;

**Текущий квартал**

-- Первый день квартала

select trunc (SYSDATE, 'Q') from dual;

-- Последний день квартала

select add\_months(trunc(sysdate,'q'),3)-1 from dual;

**Прошлый квартал**

-- Первый день прошлого квартала

select trunc(add\_months(sysdate,-3),'q') from dual;

-- Последний день прошлого квартала

select add\_months(trunc(add\_months(sysdate,-3),'q'),3)-1 from dual;

**Следующий квартал**

-- Первый день следующего квартала

select add\_months(trunc(sysdate,'q'),3) from dual;

-- Последний день следующего квартала

select add\_months(trunc(add\_months(sysdate,3),'q'),3)-1 from dual;

**Текущий год**

-- Первый день года

select trunc (SYSDATE, 'Y') from dual;

-- Последний день года

select ADD\_MONTHS(trunc (SYSDATE, 'YEAR'),12)-1 FROM DUAL;

**Прошедший год**

-- Первый день прошлого года

select ADD\_MONTHS (trunc (SYSDATE, 'YEAR'), -12) FROM DUAL;

-- Последний день прошлого года

select ADD\_MONTHS (trunc (SYSDATE, 'YEAR'), -1 ) +30 FROM DUAL;

**Следующий год**

-- Первый день следующего года

select ADD\_MONTHS(trunc (SYSDATE, 'YEAR'),12) FROM DUAL;

-- Последний день следующего года

select ADD\_MONTHS(trunc (SYSDATE, 'YEAR'),24)-1 FROM DUAL;

**NVL ФУНКЦИЯ**

В этом учебном пособии вы узнаете, как использовать Oracle/PLSQL **функцию NVL** с синтаксисом и примерами.

**Описание**

Oracle/PLSQL функция NVL позволяет заменить значение, когда встречается Null значение.

**Синтаксис**

Синтаксис Oracle/PLSQL функции NVL:

NVL( string1, replace\_with )

**Параметры или аргументы**

string1 строка, которая проверяется на Null значение.

replace\_with возвращаемое значение, если string1 является Null.

**Применение**

**Функцию NVL** можно использовать в следующих версиях Oracle/PLSQL:

* Oracle 12c, Oracle 11g, Oracle 10g, Oracle 9i, Oracle 8i

**Пример**

Рассмотрим несколько примеров функции Oracle NVL и изучим, как использовать **функцию NVL** в Oracle/PLSQL.

SELECT NVL(supplier\_city, 'n/a')

FROM suppliers;

SQL запрос приведенный выше вернет 'n/a', если в поле supplier\_city содержится Null значение. В противном случае, он будет возвращать значение supplier\_city.

Другой пример использования функции NVL в Oracle/PLSQL:

SELECT supplier\_id, NVL(supplier\_desc, supplier\_name)

FROM suppliers;

SQL запрос вернет поле supplier\_name если supplier\_desc содержит Null. В противном случае, он будет возвращать supplier\_desc.

Последний пример использования функции NVL в Oracle/PLSQL:

SELECT NVL(commission, 0)

FROM sales;

SQL запрос вернет 0, если поле commission содержится Null значение. В противном случае, он вернет значение поля commission.

# DECODE ФУНКЦИЯ

В этом учебном пособии вы узнаете, как использовать Oracle/PLSQL **функцию DECODE** с синтаксисом и примерами.

## Описание

Oracle/PLSQL функция DECODE имеет функциональные возможности оператора IF-THEN-ELSE.

## Синтаксис

Синтаксис Oracle/PLSQL функции DECODE:

DECODE( expression , search , result [, search , result]... [, default] )

### Параметры или аргументы

expression - выражение для сравнения.

search - значение, которое сравнивается с expression.

result - значение, возвращаемое, если выражение expression совпало с искомым search.

default - необязательный. Если совпадений не найдено, функция DECODE вернет значение по умолчанию. Если значение по умолчанию не указано, то функция DECODE вернет NULL (если соответствий не найдено).

### Применение

**Функцию DECODE** можно использовать в следующих версиях Oracle/PLSQL:

* Oracle 12c, Oracle 11g, Oracle 10g, Oracle 9i

## Пример

Вы можете использовать функцию DECODE в SQL запросе следующим образом:

SELECT supplier\_name,

DECODE(supplier\_id, 10000, 'IBM',

10001, 'Microsoft',

10002, 'Hewlett Packard',

'Gateway') result

FROM suppliers;

Запрос с использованием DECODE эквивалентен конструкции IF-THEN-ELSE.

IF supplier\_id = 10000 THEN

result := 'IBM';

ELSIF supplier\_id = 10001 THEN

result := 'Microsoft';

ELSIF supplier\_id = 10002 THEN

result := 'Hewlett Packard';

ELSE

result := 'Gateway';

END IF;

Функция DECODE будет сравнивать каждое значение supplier\_id, одно за другим.

## Часто задаваемые вопросы

### Вопрос:

Один из наших читателей хотел узнать, как использовать функцию DECODE, чтобы сравнить две даты (то есть: date1 и date2), где date1 > date2, функция DECODE должна вернуть date2. В противном случае функция DECODE должна вернуть date1.

### Ответ:

Для этого, используйте функцию DECODE следующим образом:

DECODE((date1 - date2) - ABS(date1 - date2), 0, date2, date1)

Приведенная ниже формула будет равна 0, если date1 больше date2:

(date1 - date2) - ABS(date1 - date2)

### Полезный совет № 1:

Один из наших читателей предложил объединить функцию SIGN с функцией DECODE следующим образом:

Пример с датами выше, может быть модифицированы следующим образом:

DECODE(SIGN(date1-date2), 1, date2, date1)

Сочетание SIGN / DECODE также полезно для цифровых сравнений например продажа бонусов.

DECODE(SIGN(actual-target), -1, 'Нет бонусов для тебя', 0,'Просто сделай это', 1, 'Поздравляю, вы победитель')

### Полезный совет № 2:

Один из наших читателей предложил использовать функцию LEAST (вместо функции DECODE) следующим образом:

Пример с датами выше, может быть модифицированы следующим образом:

LEAST(date1, date2)

### Вопрос:

Я хотел бы знать, если это возможно, чтобы использовать функцию DECODE для диапазонов чисел, то есть 1-10 = 'категория 1', 11-20 = 'категорию 2', вместо того, чтобы индивидуально декодировать каждый номер.

### Ответ:

К сожалению, вы не можете использовать функцию DECODE для диапазонов чисел. Однако, вы можете попробовать создать формулу, которая будет определять один номер для одного диапазона, и другой номер для другого диапазона, и так далее.

Например:

SELECT supplier\_id,

DECODE(TRUNC ((supplier\_id - 1) / 10), 0, 'category 1',

1, 'category 2',

2, 'category 3',

'unknown') result

FROM suppliers;

Этот пример, на основе формулы:

TRUNC((supplier\_id - 1) / 10)

Формула будет оценивать 0, если supplier\_id находится между 1 и 10.  
Формула будет оценивать в 1, если supplier\_id между 11 и 20.  
Формула будет оценивать по 2, если supplier\_id между 21 и 30.

### Вопрос:

Мне нужно написать запрос DECODE, который будет возвращать следующие:

Если yrs\_of\_service <1, то вернуть 0,04 Если yrs\_of\_service> = 1 и <5, то вернуть 0,04 Если yrs\_of\_service> 5 затем вернуть 0,06

Как я могу это сделать?

### Ответ:

Вам нужно будет создать формулу, которая рассчитает единичное число для каждого из вашего диапазона.

Например:

SELECT emp\_name,

DECODE(TRUNC ((yrs\_of\_service + 3) / 4), 0, 0.04,

1, 0.04,

0.06) as perc\_value

### FROM employees;

### Вопрос:

Есть ли ограничение на количество аргументов, которые вы можете указать в одном операторе DECODE? Я получаю сообщение об ошибке "ORA-00939: слишком много аргументов для функции".

### Ответ:

Да, максимальное количество компонентов, которые вы можете иметь в функции DECODE 255. Это включая expression, search и result аргументы.

# CASE ОПЕРАТОР

В этом учебном пособии вы узнаете, как использовать оператор **CASE** в Oracle/PLSQL c синтаксисом и примерами.

## Описание

В Oracle/PLSQL оператор **CASE** имеет функциональность IF-THEN-ELSE. Начиная с Oracle 9i, вы можете использовать оператор CASE в SQL предложении.

## Синтаксис

Синтаксис оператора CASE в Oracle/PLSQL:

CASE [ expression ]

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2

...

WHEN condition\_n THEN result\_n

ELSE result

END

### Параметры или аргументы

**expression**

Не является обязательным. Это значение, которое вы сравниваете с условиями (то есть: condition\_1 ... condition\_n).

**condition\_1 .. condition\_n**

должны быть одного типа. Условия оцениваются по порядку, одно за другим. После того, как условие примет значение TRUE (истина), оператор **CASE** вернет результат, и не будет оценивать условия дальше.

**result\_1 .. result\_n**

все должны быть одного типа данных. Это значение возвращается единожды, когда condition примет TRUE (истина).

### Примечание

* Если условие не примет TRUE, то оператор CASE вернет значение предложения ELSE.
* Если предложение ELSE опущено и условие не примет TRUE, то оператор CASE вернет NULL.
* Оператор CASE может иметь до 255 сравнений. Каждое предложение WHEN ... THEN рассматривает 2 сравнения.

### Применение

**Оператор CASE** можно использовать в следующих версиях Oracle / PLSQL:

* Oracle 12c, Oracle 11g, Oracle 10g, Oracle 9i

## Пример

Вы можете использовать оператор CASE в SQL предложении следующим образом:

SELECT table\_name,

CASE owner

WHEN 'SYS' THEN 'The owner is SYS'

WHEN 'SYSTEM' THEN 'The owner is SYSTEM'

ELSE 'The owner is another value'

END

FROM all\_tables;

Или вы могли бы написать SQL запрос, используя оператор CASE следующим образом: (опустив expression в предложение WHEN .. THEN). Например:

SELECT table\_name,

CASE

WHEN owner='SYS' THEN 'The owner is SYS'

WHEN owner='SYSTEM' THEN 'The owner is SYSTEM'

ELSE 'The owner is another value'

END

FROM all\_tables;

Эти два примера оператора CASE эквивалентны следующему условному оператору IF-THEN-ELSE:

IF owner = 'SYS' THEN

result := 'The owner is SYS';

ELSIF owner = 'SYSTEM' THEN

result := 'The owner is SYSTEM'';

ELSE

result := 'The owner is another value';

END IF;

Оператор CASE будет сравнивать каждое значение owner, одно за другим.

Предложение ELSE в операторе CASE не является обязательным. Его можно опустить. Давайте посмотрим предыдущий SQL запрос с опущенным предложением ELSE.

SQL запрос будет выглядеть следующим образом:

SELECT table\_name,

CASE owner

WHEN 'SYS' THEN 'The owner is SYS'

WHEN 'SYSTEM' THEN 'The owner is SYSTEM'

END

FROM all\_tables;

С отсутствующим предложением ELSE, если ни одно из условий не приняло TRUE, оператор CASE вернет NULL.

### Сравнивая 2 условия

Вот пример, который показывает, как использовать оператор CASE, чтобы сравнить различные условия:

SELECT

CASE

WHEN a < b THEN 'hello'

WHEN d < e THEN 'goodbye'

END

FROM suppliers;

Сравнение 2-х полей в одном предложении CASE:

SELECT supplier\_id,

CASE

WHEN supplier\_name = 'IBM' and supplier\_type = 'Hardware' THEN 'North office'

WHEN supplier\_name = 'IBM' and supplier\_type = 'Software' THEN 'South office'

END

FROM suppliers;

Итак, если поле supplier\_name = 'IBM' и поле supplier\_type = 'Hardware', то оператор CASE вернет 'North office'. Если поле supplier\_name = 'IBM' и supplier\_type = 'Software', оператор CASE вернет 'South office'.