**Использование однострочных функций для получения требуемых выходных данных.**

Однострочные функции:

· Управляют элементами данных

· Принимают аргументы и возвращают одно значение

· Работают с каждой записью, возвращаемой запросом

· ***Возвращают один результат на строку***

· Могут изменять тип данных

· Могут быть вложенными

· Принимают аргументы, которые могут быть полями или выражениями

***Синтаксис:***

Function\_name [(arg1,arg2,…)]

***Виды однострочных функций:***

· *Общие*

· *Символьные*

o *Функции преобразования регистра символов (LOWER, UPPER, INITCAP)*

o *Функции манипулирования символами ( CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM, REPLACE)*

· *Числовые (ROUND,TRUNC,MOD)*

· *Работа с датами*

· *Преобразования*

***Функции преобразования регистра символов***

Преобразуют регистр символьных строк

*1.* *LOWER (‘SQL Course’)*  **⇒** *sql course*

*2.* *UPPER (‘SQL Course’)*  **⇒** *SQL COURSE*

*3.* *INITCAP (‘SQL Course’)*  **⇒** *Sql Course*

***Пример:***Вывод номера сотрудника Higgins, его фамилии и номера отдела

SELECT employee\_id, last\_name, department\_id

FROM employee

WHERE LOWER(last\_name) = ‘higgins’;

***Функции манипулирования символами***

**1.** **CONCAT (‘Hello’,’World’) ⇒ HelloWorld**

**2.** **SUBSTR (‘HelloWorld’,1,5) ⇒ Hello (2 аргумента после строки: с какого элемента и сколько элементов)**

**3.** **LENGTH (‘HelloWorld’) ⇒ 10**

**4.** **INSTR (‘HelloWorld’,’W’) ⇒ 6 (определяет нахождение буквы)**

**5.** **LPAD (salary,10,’\*’) ⇒ \*\*\*\*\*24000 ( добавляет с левой части строки определенный набор символов (при не нулевом первом аргументе - строке)**

**6.** **RPAD – аналогично LPAD, только с правой стороны**

**7.** **REPLACE (‘JACK and JUE’,’J’,’BL’) ⇒ BLACK and BLUE**

**8.** **TRIM (‘H’ FROM ‘HelloWorld’) ⇒ elloWorld**

***Числовые***

**1.** ROUND : округляет значение до заданного количества десятичных знаков

ROUND (45.926, 2)  **⇒** 45.93

ROUND (45.926, 0)  **⇒** 45

ROUND (45.926, -1)  **⇒** 50

SELECT ROUND (45.926, -1)

FROM DUAL ;

*(DUAL – фиктивная таблица, используемая для получения результатов выполнения функций и вычислений)*

**2.** TRUNC : усекает значение до заданного количества десятичных знаков

TRUNC (45.926, 2)  **⇒** 45.92

TRUNC (45.926)  **⇒** 45

TRUNC (45.926, -1)  **⇒** 40

**3.** MOD : возвращает остаток от деления

MOD (1600, 300) **⇒**100

***Работа с датами***

· По умолчанию дата выдается в формате DD-MON-RR (ЧИСЛО-МЕСЯЦ-ГОД)

**1.** **SYSDATE** – функция, которая возвращает: дату, время

**2.** Арифметические операции с датами:

· Результатом прибавления числа к дате и вычитания числа из даты является дата

· Результатом вычитания одной даты из другой является количество дней, разделяющих эти даты

· Прибавление часов к дате производится путем деления количества часов на 24

**3.** **MONTHS\_BETWEEN** – количество месяцев между двумя датами

MONTHS\_BETWEEN (’01-SEP-95’, ’11-JAN-94’) **⇒** 19.6774194

**4.** **ADD\_MONTHS** – добавление календарного месяца к дате

ADD\_MONTHS (’11-JAN-94’, 6)  **⇒**‘11-JUL-94’

**5.**  **NEXT\_DAY** – следующий день от определенной даты

NEXT\_DAY (’01-SEP-95’, ‘FRIDAY’) **⇒** ’08-SEP-95’

**6.** **LAST\_DAY** – последний день месяца

LAST\_DAY (’01-FEB-95’)  **⇒** ’28-FEB-95’

**7.** **ROUND** - округление дат

(SYSDATE=’25-JUL-03’)

ROUND(SYSDATE, ‘MONTH’)  **⇒** 01-AUG-03

ROUND(SYSDATE, ‘YEAR’) **⇒** 01-JAN-04

**8.**  **TRUNC** – отсечение дат

(SYSDATE=’25-JUL-03’)

TRUNC(SYSDATE, ‘MONTH’)  **⇒** 01-JUL-03

TRUNC(SYSDATE, ‘YEAR’) **⇒** 01-JAN-03

***Функции Преобразования***

· *Неявное преобразование типа данных*

Для операций присваивания Oracle может автоматически выполнять следующие преобразования:

**1.** Из VARCHAR2,CHAR  **⇒** NUMBER

**2.** VARCHAR2/CHAR **⇒** DATE

**3.** NUMBER **⇒** VARCHAR2

**4.** DATE  **⇒**VARCHAR2

· *Явное преобразование типов данных с помощью функций :*

**1.** **TO\_CHAR (date, ‘format\_model’) – с датами**

-различает символы верхнего и нижнего регистров

-может включать любые разрешенные элементы формата даты

-использует элемент fm для удаления конечных пробелов и ведущих нулей

-отделяется от значения даты запятой

Элементы модели формата даты:

o YYYY – полный год цифрами

o YEAR – год прописью (на английском)

o MM – двузначное цифровое обозначение месяца

o MONTH – полное название месяца

o MON – трехзначное буквенное сокращение месяца

o DY – трехзначное буквенное сокращение дня недели

o DAY – полное название дня недели

o DD – номер дня месяца

o HH24:MI:SS AM è 15:45:32 PM

o DD “of” MONTH è 12 of OCTOBER

**2.** **TO\_CHAR (number, ‘format\_model’) – с числами**

Элементы модели формата числа:

o 9 – цифра

o 0 – ввод нуля

o $ - плавающий знак доллара

o . – вывод десятичной точки

o , - вывод разделителя троек цифр

TO\_CHAR(salary, ‘$99,999.00’ **⇒** $8,000.00

**3.** **TO\_NUMBER (char [, ‘format\_model’])**

Преобразование символьной строки в числовой формат с использованием функции

SELECT TO\_NUMBER('1242.45', '9999.99')

FROM DUAL;

--Результат: 1242,45

Format\_model – необязательный параметр

|  |
| --- |
| SELECT TO\_NUMBER ('1242.45')  FROM DUAL;    --Результат: 1242,45 |

**4.** **TO\_DATE (char [, ‘format\_model’])**

Преобразование символьной строки в формат даты с использованием функции

SELECT TO\_DATE('2014/07/22', 'yyyy/mm/dd')

FROM DUAL;

--Результат: 22.07.2014

SELECT TO\_DATE('30.01.2019 18:30:52', 'DD.MM.YYYY HH24:MI:SS') FROM DUAL;

--Результат: 30.01.2019 18:30:52

***Вложенные функции***

· Однострочные функции могут быть вложены на любую глубину

· Вложенные функции вычисляются от самого глубокого уровня к внешнему

F3(F2(F1(col,arg1), arg2), arg3)

Пример: last\_name = Hunold

SELECT UPPER(CONCAT(SUBSTR(LAST\_NAME, 1, 8), ‘\_US’)) è **⇒** HUNOLD\_US

***Общие функции***

o Общие функции работают с любыми типами данных и обрабатывают неопределенные значения (NVL, NVL2, NULLIF, COALESCE)

**1.** **NVL** - преобразует неопределенное число в действительное

o Используемые типы данных – DATE, CHAR, NUMBER

o Типы данных должны совпадать

**2.** **NVL2** - расширяет функциональность функции NVL. Это позволяет заменяет значение, когда встречается Null значение, а также когда встречается не-Null значение.

o NVL2( string1, value\_if\_NOT\_null, value\_if\_null )

### ***Параметры или аргументы***

string1 строка для проверки на Null значение.

value\_if\_NOT\_null это значение возвращается, если string1 имеет не Null значение.

value\_if\_null это значение возвращается, если string1 имеет Null значение.

**3.** **NULLIF** -сравнивает expr1 и expr2. Если expr1 и expr2 равны, функция NULLIF возвращает NULL. В противном случае, она возвращает expr1.

SELECT NULLIF(12, 12) AS RESULT FROM DUAL;

--Результат: NULL

SELECT NULLIF(12, 13) AS RESULT FROM DUAL;

--Результат: 12

o

**4.** COALESCE ( expr1, expr2, … expr\_n ) возвращает первое ненулевое выражение из списка. Если все выражения определены как Null, то функция COALESCE вернет Null.

***Условные выражения***

**1.** **CASE (метод**) - имеет функциональность [IF-THEN-ELSE](https://oracleplsql.ru/if-then-else.html)

CASE [ expression ]

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2

…

WHEN condition\_n THEN result\_n

ELSE result

END

Пример:

|  |
| --- |
| 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; |

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.

**2.** **DECODE (функция)**

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).

***Пример:***

SELECT supplier\_name,

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

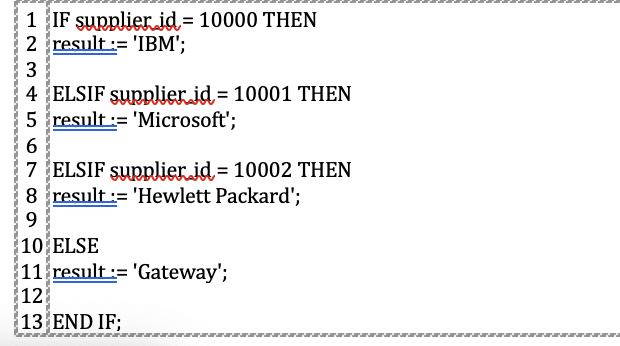
10001, 'Microsoft',

10002, 'Hewlett Packard',

'Gateway') result

FROM suppliers;

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



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