

## Общие функции

Эти функции работают с любыми типами данных и обрабатывают неопределенные значения:

- NVL (выражение1, выражение2)
- NVL2 (выражение1, выражение2, выражение3)
- NULLIF (выражение1, выражение2)
- COALESCE (выражение1, выражение2, ..., выражениеN)

- NVL – преобразует неопределенное значение в действительное
- NVL2 – Если **Выражение1** определено (is not null), NVL2 возвратит **выражение2**. Если **Выражение1** не определено (is null), NVL2 возвратит **выражение2**. Аргумент выражения1 может быть любого типа.
- NULLIF – сравнивает два выражения и возвращает неопределенное значение (Null), если выражения равны, или возвращает первое выражение в противном случае.
- Coalesce – возвращает первое определенное значение из списка выражения.

## Функция NVL

Преобразует неопределенное значение в действительное:

- Используемые типы данных – DATE, символьные (CHARACTER) и числовые (NUMBER).
- Типы данных должны совпадать:
  - NVL(commission\_pct, 0)
  - NVL(hire\_date, '01-JAN-97')
  - NVL(job\_id, 'No Job Yet')

- Функция NVL используется для преобразования неопределенного значения (NULL) в действительное.

Синтаксис

NVL(выражение1, выражение2)

**Выражение1** – исходное значение или выражение, которое может содержать неопределённое значение.

**Выражение2** – конечное значение для преобразования неопределенного значения.

NVL – используется для преобразования любого типа, но тип возвращаемых данных всегда такой как у **выражения1**

NUMBER –

NVL(числовой\_столбец, 9)

DATE –

NVL(столбец\_даты, '01-JAN-95')

CHAR или VARCHAR2 –

NVL(символьный\_столбец, 'Unavailable')

## Использование функции NVL

Для вычисления годового дохода служащих, необходимо оклад умножить на 12, а затем прибавить сумму комиссионных.

Только для служащих с комиссионными, а у кого не определено значение комиссионных используем NVL.

```
SELECT last name, salary, NVL(commission pct, 0) AS COM_PCT,  
       (salary*12) + (salary*12*NVL(commission pct, 0)) AS AN_SAL  
FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...  
20 rows selected

## Использование функции NVL2

```
SELECT last name, salary, commission_pct  
      NVL2(commission_pct,  
           'SAL+COMM', 'SAL') income  
FROM employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	2	SAL+COMM
Abel	11000	3	SAL+COMM
Taylor	8600	2	SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected

- Функция NVL2 проверяет первое выражение. Если оно определено, тогда функция NVL2 возвращает второе выражение. Если первое выражение не определено, результатом работы будет третье выражение.

NVL2(выражение1, выражение2, выражение3)

**Выражение1** – исходное значение или выражение

**Выражение2** – возвращает значение, если выражение1 определено.

**Выражение3** – возвращает значение, если выражение не определено.

В примере проверка значения `commission_pct`. Если есть значение, то `'sal+comm'`. Если нет, то `'sal'`.

Тип выражения 2 и 3 может быть любого типа, кроме LONG. Если типы данных выражения 2 и 3 различные, то сервер выражения3 преобразует в тип выражения2. Здесь преобразование не требуется.

## Использование функции NULLIF

```
SELECT first_name, LENGTH(first_name) "expr1",  
       last_name, LENGTH(last_name) "expr2",  
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result  
FROM employees;
```

Diagram labels: 1 points to the first expression `LENGTH(first_name)`; 2 points to the second expression `LENGTH(last_name)`; 3 points to the `NULLIF` function call.

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	
Diana	5	Lorentz	7	5
Kevin	5	Mourgos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	

Diagram labels: 1 points to the `expr1` column; 2 points to the `expr2` column; 3 points to the `RESULT` column.

...  
20 rows selected

ORACLE

- Функция NULLIF сравнивает два выражения. Если они равны, функция возвращает неопределенное значение, если нет — первое выражение NULLIF(выражение1, выражение2)

Выражение1 — исходное значение сравниваемое с выражением2.

Выражение2 — исходное значение сравниваемое с выражением1.

## Использование функции COALESCE

- Преимущество функции COALESCE по сравнению с функцией NVL состоит в том, что функция COALESCE может обрабатывать несколько альтернативных значений.
- Если первое выражение определено, функция возвращает это выражение; в противном случае она проверяет оставшиеся выражения

- Функция COALESCE возвращает первое определенное выражение в списке.

COALESCE(выражение1, выражение2, ... выражениеN)

Выражение1 - возвращаемое выражение, если оно имеет определенное значение.

Выражение2 – возвращаемое выражение, если первое выражение не определено, а это выражение имеет определенное значение.

выражениеN – возвращаемое выражение, если предыдущее выражение не определено.

## Использование функции COALESCE

```
SELECT last name,  
       COALESCE (manager id, commission pct, -1) comm  
FROM   employees  
ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	149
Zlotkey	100
Taylor	149
Abel	149
King	-1
Kochhar	100
De Haan	100

...

20 rows selected

- В примере выводится табельный номер менеджера, если он определен. Если табельный номер менеджера не задан выводится commission\_pct, если они определены. Если ни табельный номер менеджера, ни комиссионные не установлены, выводится -1.

# Условные выражения

- **Позволяют применять логические конструкции ЕСЛИ-ТО-ИНАЧЕ (IF-THEN-ELSE) внутри команды SQL**
- **Два метода:**
  - **выражение CASE**
  - **функция DECODE**



## Выражение CASE

Помогает создавать условные запросы, которые выполняют действия логического оператора IF-THEN-ELSE:

```
CASE выражение
  WHEN сравн_выражение1 THEN возвр_выражение1
  [WHEN сравн_выражение2 THEN возвр_выражение2
  WHEN сравн_выражениеn THEN возвр_выражениеn
  ELSE else_выражение]
END
```

Выражение CASE позволяет производить логическую обработку оператора IF-THEN-ELSE в командах SQL, не вызывая процедуры.

В простом выражении CASE сервер Oracle ищет первую пару WHEN ... THEN, в которой совпадают выражение и **сравн\_выражение** и возвращает **возвр\_выражение**. Если нет совпадений ни в одном из сравниваемых пар WHEN... THEN и есть предложение ELSE, возвращается **else\_выражение**. Если такого предложения ELSE нет, возвращается неопределённое значение. Нельзя задавать литерал NULL для возвр\_выражений и else\_выражений. Все возвращаемые значения должны быть одного типа.

# Использование выражения CASE

Помогает создавать условные запросы, которые выполняют действия логического оператора IF-THEN-ELSE:

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                 WHEN 'ST_CLERK' THEN 1.15*salary  
                 WHEN 'SA_REP' THEN 1.20*salary  
       ELSE salary END "REVISED SALARY"  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

- В приведенной команде SQL расшифровывается значение идентификатора должности JOB\_ID. Если значение JOB\_ID совпадает с IT\_PROG, оклад повышается на 10%; если JOB\_ID равно ST\_CLERK, повышение на 15%; если JOB\_ID равно SA\_REP, оклад увеличивается на 20%. Для всех остальных должностей оклад не изменяется.
- Такую же команду можно написать с помощью функции DECODE.

## Функция DECODE

Помогает создавать условные запросы, которые выполняют действия логического условия CASE или оператора IF-THEN-ELSE:

```
DECODE (столбец|выражение, вариант1, результат1  
        [, вариант2, результат2,...,]  
        [, результат_по_умолчанию])
```

- Функция DECODE действует подобно IF-THEN-ELSE в различных языках. Функция DECODE расшифровывает столбец или выражение после сравнения его с каждым искомым значением варианта. Если выражение равно искомому значению, функция возвращает соответствующий результат.

# Использование функции DECODE

```
SELECT last name, job id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
               'ST_CLERK', 1.15*salary,  
               'SA_REP', 1.20*salary,  
               salary)  
       REVISED_SALARY  
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

# Использование функции DECODE

Показать ставку налога на заработную плату для сотрудников 80 отдела:

```
SELECT last name, salary,  
       DECODE (TRUNC(salary/2000, 0),  
              0, 0.00,  
              1, 0.09,  
              2, 0.20,  
              3, 0.30,  
              4, 0.40,  
              5, 0.42,  
              6, 0.44,  
              0.45) TAX_RATE  
FROM   employees  
WHERE  department_id = 80;
```

# Итоги

**С помощью функций осуществляются:**

- **Вычисления с данными**
- **Изменение отдельных элементов данных**
- **Манипулирование выводом групп строк**
- **Изменение форматов дат для вывода**
- **Преобразование формата данных столбцов**
- **Обработка неопределенных значений**
- **Логическая обработка IF-THEN-ELSE**

## Обзор практического занятия 3 , часть 2

- **Составление запросов, требующих использования числовых, символьных функций и функций для работы с датами.**
- **Использование конкатенации с функциями.**
- **Составление запросов, нечувствительных к регистру символов, для проверки полезности символьных функций.**
- **Вычисление продолжительности работы служащего в месяцах и годах.**
- **Определение даты аттестации служащего**