«Базы данных» SELECT. Лекция 6

Савченко Наталья Александровна

Базовая команда SELECT

```
SELECT      *|{ [DISTINCT]      column| expression [alias], …}
FROM        table;
```

- SELECT указывает, *какие* столбцы;
- FROM указывает, из *какой* таблицы.

Синтаксис:

SELECT	список из одного или более столбцов
*	выбирает все столбцы
DISTINCT	устраняет дубликаты
столбец|выражение	выбирает заданный столбец или выражение
псевдоним	присваивает заданным столбцам другие имена
FROM таблица	указывает таблицу, содержащую столбцы

Выбор всех столбцов

```
SELECT      *
FROM        departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Выбор конкретных столбцов

```
SELECT      department_id, location_id
FROM        departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

Неопределенное значение (NULL)

Неопределенное значение (NULL) – это значение, которое недоступно, не присвоено, неизвестно или неприменимо. Это не ноль и не пробел.

```
SELECT      last_name, job_id, salary, commission_pct
FROM        employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
[illegible]	[illegible]	[illegible]	
Gietz	AC_ACCOUNT	8300	

Если в строке отсутствует значение какого-либо столбца, считается, что столбец содержит NULL.

Неопределенные значения допускаются в столбцах с данными любого типа за исключением случаев, когда столбец был создан с ограничением NOT NULL или PRIMARY KEY.

Использование псевдонима (алиаса) столбца

```
SELECT    last_name AS name, commission_pct comm
FROM      employees;
```

NAME	COMM
King	
Kochhar	
Higgins	
Gietz	

```
SELECT    last_name “Name”, salary*12 “Annual Salary”
FROM      employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
Higgins	144000
Gietz	99600

Устранение строк-дубликатов

Дубликаты устраняются с помощью ключевого слова DISTINCT в команде SELECT.

```
SELECT    DISTINCT department_id
FROM      employees;
```

DEPARTMENT_ID
10
20
50
60
80
90
110

8 rows selected.

Ограничение количества выбираемых строк

Количество возвращаемых строк можно ограничить с помощью предложения WHERE.

```
SELECT      *|{ [DISTINCT] column/expression [alias],…}
FROM        table
[WHERE   condition (s)];
```

Предложение WHERE следует за предложением FROM.

Синтаксис:

WHERE	ограничивает количество выбираемых строк, задавая условие выборки
условие	условие, состоящее из имен столбцов, выражений, констант, оператора сравнения

Предложение WHERE может сравнивать значения в столбцах, литералы, арифметические выражения, функции.
Предложение WHERE состоит из трех элементов: имя столбца; оператор сравнения; имя столбца, константа или список значений.

Операторы сравнения

Оператор	Значение
=	***Равно***
>	***Больше, чем***
>=	***Больше или равно***
<	***Меньше, чем***
<=	***Меньше или равно***
<>	***Не равно***

WHERE выражение оператор значение

Примеры:

```
... WHERE hire_date='01-JAN-95'
... WHERE salary>=6000
... WHERE last_name='Smith'
```

Псевдонимы не могут использоваться в предложении WHERE.
Символы != и ^= могут также применяться для проверки условия «не равно».

Другие условия сравнения

Оператор	Значение
BETWEEN…AND …	***Находится в диапазоне от одного значения до другого (включительно)***
IN(список)	***Совпадает с каким-либо значением списка***
LIKE	***Соответствует символьному шаблону***
IS NULL	***Является неопределенным значением***

Использование условия BETWEEN

Условие BETWEEN используется для вывода строк на основе диапазона значений

```
SELECT      last_name, salary
FROM        employees
WHERE       salary BETWEEN 2500 AND 3500;
```

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Использование условия IN

Условие принадлежности IN используется для проверки на вхождение значений в список.

```
SELECT      employee_id, last_name, salary, manager_id
FROM        employees
WHERE       manager_id IN (100, 101, 201);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

Условие IN может использоваться с данными любого типа. Если в список входят символьные строки и даты, они должны быть заключены в апострофы (‘ ’)

Использование условия LIKE

Условие LIKE используется для поиска символьных значений по шаблону с метасимволами. Условия поискам могут включать алфавитные и цифровые символы: «%» - обозначает ноль или много символов, «_» - обозначает один символ.

```
SELECT      first_name
FROM        employees
WHERE       first_name LIKE  'S%';
```

```
SELECT      last_name, hire_date
FROM        employees
WHERE       hire_date LIKE  '%95';
```

Логические условия

Оператор	Значение
AND	***Возвращает результат ИСТИННО, если выполняются оба условия.***
OR	***Возвращает результат ИСТИННО, если выполняется любое из условий.***
NOT	***Возвращает результат ИСТИННО, если следующее условие не выполняется.***

Приоритеты операторов

Порядок вычисления	Оператор
1	Арифметические операторы
2	Операторы конкатенации
3	Операторы сравнения
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT
7	AND
8	OR

Изменить стандартную последовательность можно с помощью круглых скобок, в которые заключаются выражения обрабатываемые первыми.

Предложение ORDER BY

Предложение ORDER BY используется для сортировки строк. В команде SELECT предложение ORDER BY указывается последним.

```
SELECT      last_name, job_id, department_id, hire_date
FROM        employees
ORDER BY  hire_date;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

```
ORDER BY  (столбец, выражение) [ASC|DESC]
```

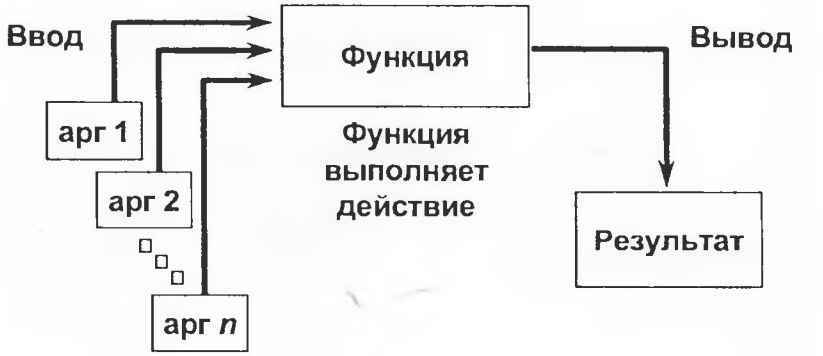
Синтаксис:

ORDER BY задает порядок вывода выбранных строк
ASC упорядочивает строки в порядке возрастания (по умолчанию)
DESC упорядочивает строки в порядке убывания

Функции SQL

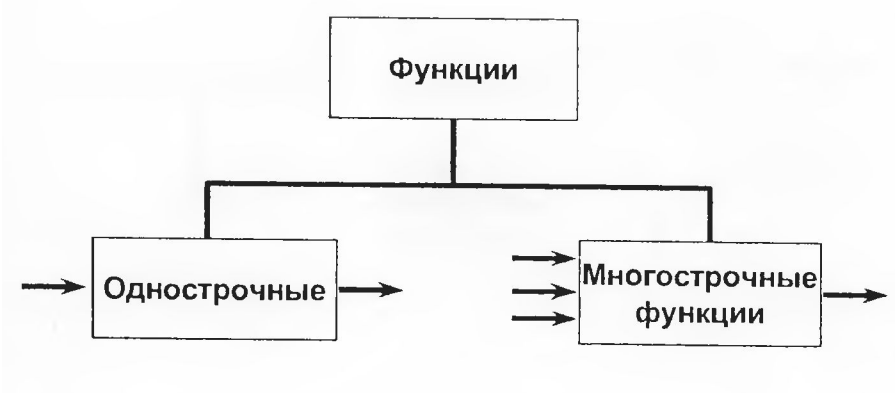
Функции являются очень мощным средством SQL и используются в следующих целях:

- Вычисления над данными;
- изменение отдельных единиц данных;
- управление выводом групп строк;
- форматирование чисел и дат для вывода;
- преобразование типов данных.



Функции SQL принимают один или несколько аргументов и всегда возвращают значение.

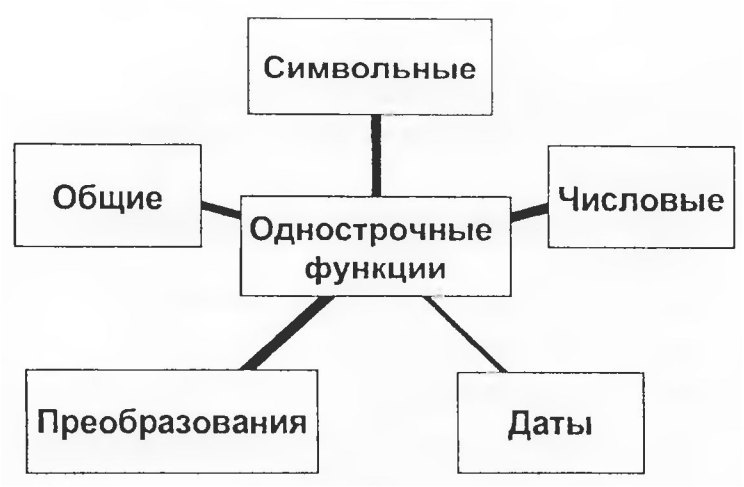
Два типа функций SQL



Однострочные функции
Эти функции работают только с одной строкой и возвращают по одному результату для каждой строки. Однострочные функции могут быть разных типов (например: символьные, числовые, для работы с датами, функции преобразования).

Многострочные функции
Эти функции работают с группой строк и выдают по одному результату для каждой группы. Их часто называют групповыми функциями.

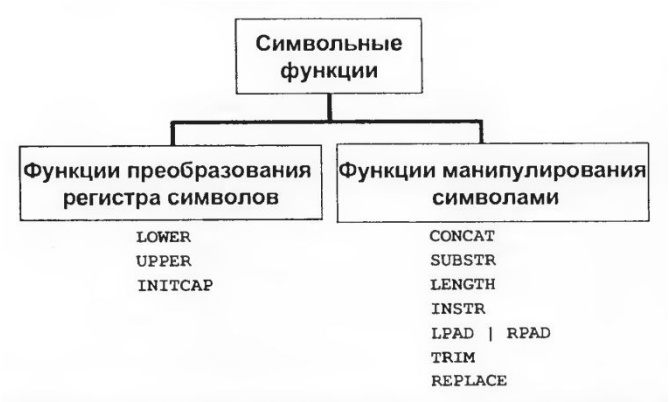
Однострочные функции



Символьные функции: принимают на входе символьные данные, а возвращают как символьные, так и числовые значения.
Числовые функции: принимают на входе числовые данные и возвращают числовые значения.
Функции преображения: преобразуют значение из одного типа данных в другой.

Функции для обработки дат: работают с значениями типа DATE. Все функции для работы с датами возвращают значение типа DATE за исключением функции MONTH_BETWEEN, которая возвращает число.
Общие функции: NVL, NVL2, NULLIF, COALSECE, CASE, DECODE.

Символьные функции (1)



Столбец	Назначение
LOWER (столбец|выражение)	***Преобразует алфавитные символы в нижний регистр***
UPPER (столбец|выражение)	***Преобразует алфавитные символы в верхний регистр***
INITCAP (столбец|выражение)	***Преобразует символьные значения: первая буква каждого слова становится заглавной, остальные - строчные.***
CONCAT (столбец1|выражение1, столбец2|выражение2)	***Присоединяет первое символьное значение ко второму. Эквивалентно оператору конкатенации (|)***
SUBSTR (столбец|выражение,m[,n])	***Возвращает n символов значения, начиная с символа m. Если m отрицательно, отсчет начинается с конца символьного значения. Если n отсутствует, возвращаются все символы до конца строки.***

Символьные функции (2)

Функция	Назначение
LENGTH (столбец|выражение)	*Возвращает количество символов в значении параметра*
INSTR (столбец|выражение,' строка', [,m], [n])	*Возвращает номер позиции указанной строки в символьном значении первого параметра. Дополнительно можно задать позицию m начала поиска в первом параметре и число обнаружений n строки По умолчанию m и n равны 1, что означает выполнение поиска в первом параметре, начиная с первой позиции до первого обнаружения.*
LPAD (столбец|выражение, n, 'строка') *RPAD (столбец|выражение, n, 'строка')*	*Дополняет символьное значение первого параметра слева до длины n заданными символами строки. Дополняет символьное значение первого параметра справа до длины n заданными символами строки.*
TRIM (leading|trailing|both, удаляемый_символ FROM исходная строка)	*Позволяет вырезать из исходной_строки начальные (leading), конечные (trailing) символы или и те, и другие (both). Если удаляемый_символ или исходная_строка являются символьными литералами, то их нужно заключить в апострофы.*
REPLACE(текст, искомая_строка, заменяющая строка)	*Выполняется поиск искомой_строки по текстовому значению первого параметра и в случае обнаружения производиться ее замена на заменяющую_строку.*

Функции манипулирования символами

Функция	Результат	
CONCAT ('Hello', 'World')	*HelloWorld*	*CONCAT:соединяет значения. Для функции CONCAT можно использовать не более двух параметров.*
SUBSTR ('HelloWorld', 1,5)	*Hello*	*SUBSTR: возвращает подстроку заданной длинны.*
LENGTH ('HelloWorld')	*10*	*LENGTH: возвращает длину строки в виде числового значения.*
INSTR ('HelloWorld', 'W')	*6*	*INSTR: возвращает номер позиции указанного символа.*
LPAD (salary, 10, '')*	******24000*	*LPAD: дополняет символьное значение, выровненное справа, до заданной длины.*
RPAD (salary, 10, '')*	*24000******	*RPAD:дополняет символьное значение, выровненное слева, до заданной длины.*
TRIM ('H' FROM 'HelloWorld')	*elloWorld*	*TRIM: удаляет из символьной строки начальные и/или конечные символы*

Числовые функции

Числовые функции принимают на входе числовые данные и возвращают числовые значения.

Функция	Назначение	Пример
ROUND (столбец|выражение, n)	***Округляет столбец, выражение или значение до n десятичных разрядов, а если n опущено, то до целого. Если n отрицательно, округляются разряды слева от десятичной точки.***	***ROUND (45.926, 2) 45.93***
TRUNC (столбец|выражение, n)	***Усекается столбец, выражение или значение до n десятичных разрядов, а если n опущено, то до целого. Если n отрицательно, усекаются разряды слева от десятичной точки.***	***TRUNC (45.926, 2) 45.92***
MOD (m, n)	***Возвращает остаток от деления m на n.***	***MOD (1600, 300) 100***

Работа с датами

SYSDATE-эта функция, которая возвращает:

- **дату**
- **время**

Вы можете использовать SYSDATE также, как любое другое имя столбца. Например, можно вынести текущую дату при выполнении запроса из таблицы. Обычно выполняют выбор SYSDATE из DUAL.

SYSDATE
08-MAR-01

Пример

Вывод текущей даты с использованием таблицы DUAL.

SELECT SYSDATE

Арифметические операции с датами

Т.к. в базе данных даты хранится в виде чисел, с ними можно выполнять такие арифметические операции, как сложение и вычитание. Прибавлять и вычитать можно как числовые константы, так и даты.

- 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.

Возможны следующие операции:

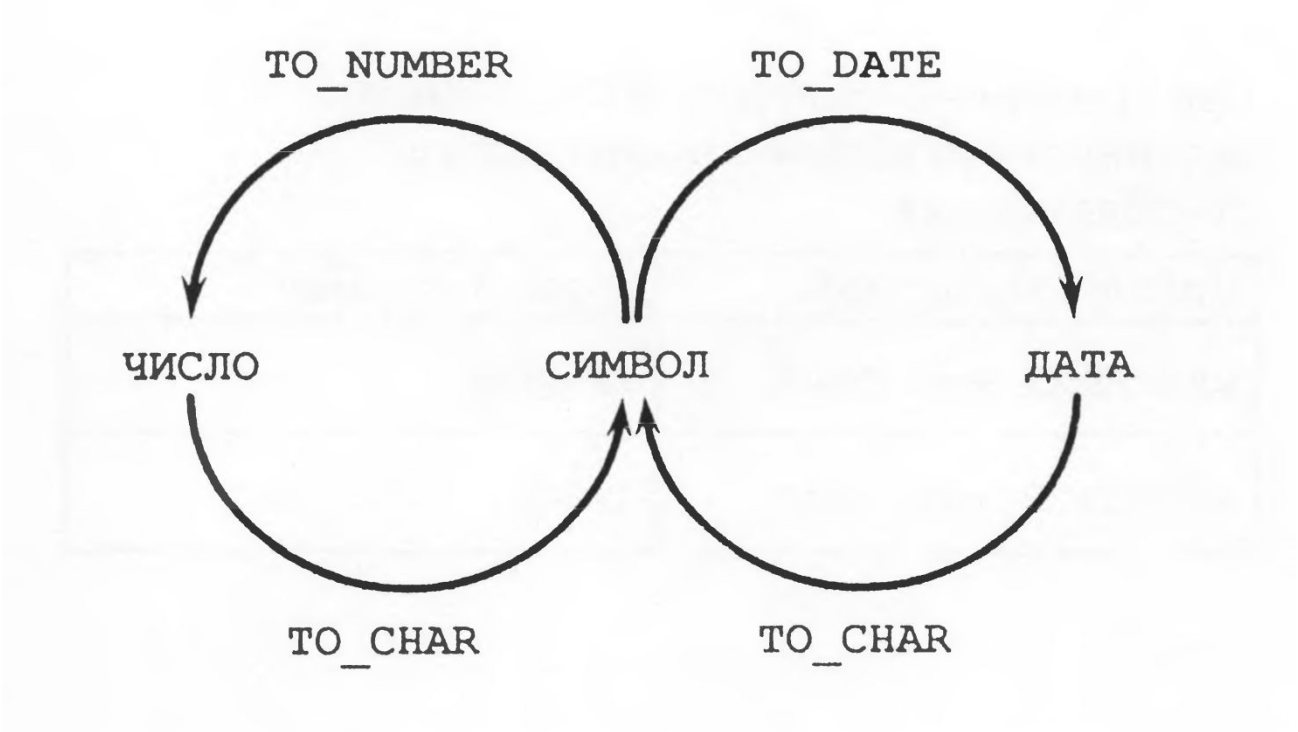
Операция	Результат	Описание
дата+число	***дата***	***Добавляет количество дней к дате***
дата-число	***дата***	***Вычитает количество дней из даты***
дата-дата	***количество дней***	***Вычитает одну дату из другой***
дата+число/24	***дата***	***Прибавляет часы к дате***

Функции для работы с датами

Функция	Описание
MONTHS_BETWEEN (date1, date2)	***Вычисляет количество месяцев между date1 и date2. Результат может быть положительным или отрицательным. Если date1 позже date2, результат положителен; если date1 предшествует date2, результат отрицателен. Дробная часть результата представляет часть месяца.***
ADD_MONTHS (date, n)	***Прибавляет n календарных месяцев к data, n должно быть целым и может быть отрицательным.***
NEXT_DAY (date, 'char')	***Возвращает дату, после параметра data, когда наступит заданный день недели ('char'); 'char' может быть числом, представляющим день недели, или строкой символов.***
LAST_DAY (date)	***Возвращает последнюю дату месяца, которому принадлежит date.***
ROUND (date [, 'fmt'])	***Возвращает дату, округленную до единицы, заданной моделью fmt. Если fmt отсутствует, дата округляется до ближайшего дня.***
TRUNC (date [, 'fmt'])	***Возвращает дату, в которой время усечено до единицы, заданной моделью fmt. Если fmt отсутствует, дата усекается до ближайшего дня.***

Явное преобразование типов данных (1)

Для преобразования значения из одного типа данных в другой SQL предлагает три функции.



Функция TO_CHAR с датами

```
TO_CHAR (date, 'format_model')
```

Модель формата:

1. Должна быть заключена в апострофы. Различает символы верхнего и нижнего регистров.
2. Может включать любые разрешенные элементы формата даты.
3. Использует элемент fm для удаления конечных пробелов и ведущих нулей.
4. Отделяется от значения даты запятой.
5. Названия дней и месяцев на выводе автоматически заполняются до нужной длины пробелами.
6. Для удаления вставленных пробелов и ведущих нулей используется элемент fm режима заполнения (fill mode).
7. Изменить ширину выходного символьного столбца можно с помощью команды COLUMN iSQL*Plus.

Элементы формата даты

YYYY	***Полный год цифрами***
YEAR	***Год прописью***
MM	***Двузначное цифровое обозначение месяца***
MONTH	***Полное название месяца***
DY	***Трехзначное алфавитное сокращенное название дня недели***
DAY	***Полное название недели***
DD	***Номер дня месяца***

Использование функций TO_CHAR с числами

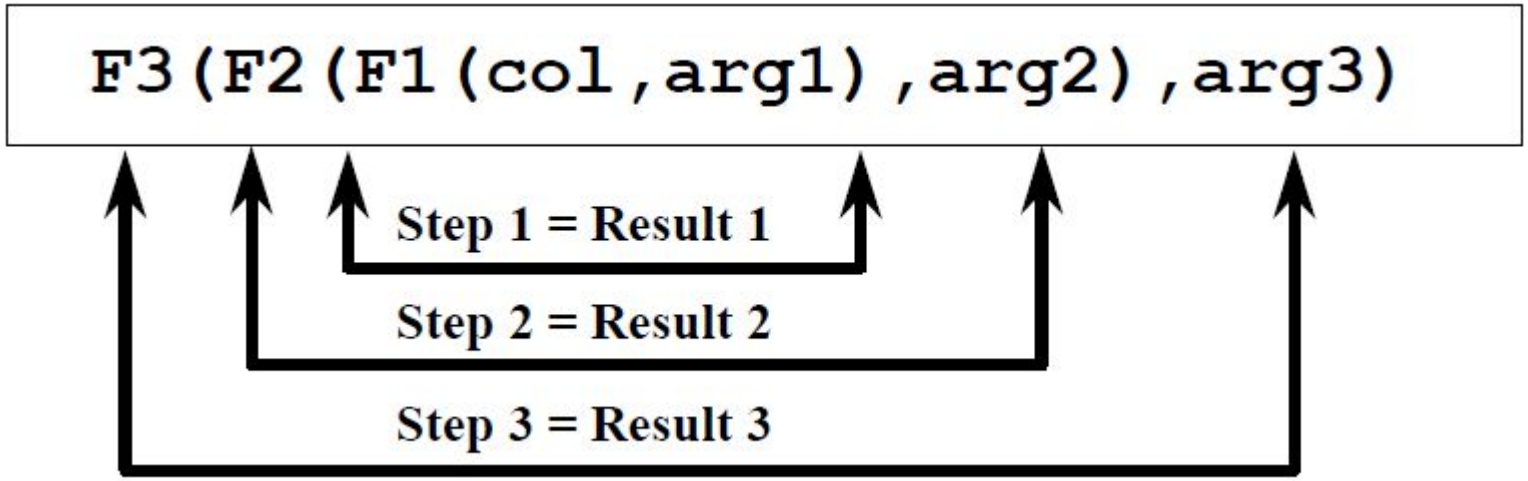
TO_CHAR (число, ‘модель_формата’)

Форматы, используемые с функцией TO_CHAR для вывода числового значения в виде символьной строки:

9	***Цифра***
0	***Вывод нуля***
$	***Плавающий знак доллара***
L	***Плавающий символ местной валюты***
.	***Вывод десятичной точки***
,	***Вывод разделителя троек цифр***

Вложенные функции

- Однострочные функции могут быть вложены на любую глубину.
- Вложенные функции вычисляются от самого глубокого уровня к внешнему.



Общие функции

Эти функции работают с любыми типами данных и используются для обработки неопределенных значений списка выражений.

Функция	***Описание***
NVL	***Преобразует неопределенное значение в действительное***
NVL2	Если выражение1 определено (is not null), NVL2 возвратит выражение2. Если выражение1 не определено (is null), NVL2 возвратит выражение 3. Аргумент выражение1 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 выражение из списка выражений.

Выражение CASE

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

```
CASE
      WHEN сравн_выражение1 THEN возвр_выражение1
      [WHEN сравн_выражение2 THEN вовзр_выражение2
       WHEN сравн_выражениеп THEN вовзр_выражениеп
      ELSE else-выражение]
END
```

Все выражения (выражение, сравн_выражение и возвр_выражение) должны быть одного типа.

Допустимые типы: CHAR, VARCHAR2, NCHAR и NVARCHAR2.

Функция DECODE

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

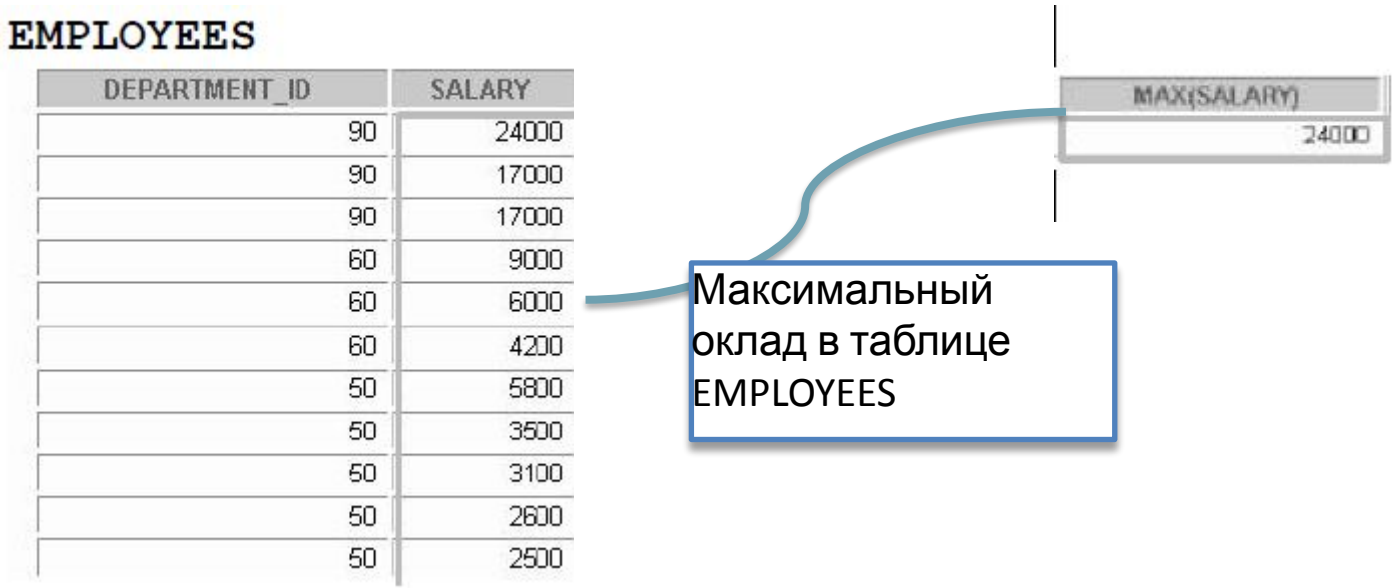
```
DECODE (столбец|выражение, вариант 1, результат 1 [ , вариант2, результат2…]
                                    [ , результат_по_умолчанию])
```

Функция DECODE расшифровывает *столбец* или *выражение* после сравнения его с каждым искомым значением варианта.

1. Если *выражение* равно искомому значению, функция возвращает соответствующий *результат*.
2. Если *выражение* не совпадает ни с одним из искомых значений, а *результат_по_умолчанию* не задан, функция возвращет неопределенное значение.

Групповые функции

Групповые функции работают с множеством строк и возвращают один результат на группу.



Типы групповых функций

Функция	Описание
AVG ([DISTINCT|ALL] n)	***Среднее значение n без учета неопределенных***
COUNT ({ *| [DISTINCT|ALL] выражение})	Количество строк, где результатом вычисления выражения является любое определенное значение. Если используется “*”, подсчитываются все выбранные строки, включая дубликаты и строки с неопределенными значениями
MAX ([DISTINCT|ALL] выражение)	***Максимальное значение выражения без учета неопределенных значений***
MIN ([DISTINCT|ALL] выражение)	Минимальное значение выражения без учета неопределенных значений
STDDEV ([DISTINCT|ALL] n)	***Стандартное отклонение значений n без учета неопределенных значений***
SUM ([DISTINCT|ALL] n)	Суммирование значений n без учета неопределенных значений
VARIANCE ([DISTINCT|ALL] n)	***Дисперсия значений n без учета неопределенных значений***

Синтаксис групповых функций

```
SELECT      [столбец,] групп_функция (столбец), …
FROM        таблица
[WHERE      условие]
[GROUP BY   столбец]
[ORDER BY столбец];
```

-- Если используется слово DISTINCT, дубликаты при вычислениях функции не учитываются. Если используется слово ALL, рассматриваются все значения, включая дубликаты. Слово ALL указывать не обязательно, т.к. оно используется по умолчанию.

-- Допустимые типы данных для аргумента: CHAR, VARCHAR2, NUMBER или DATE, если задано *выражение*.

-- Все групповые функции, кроме COUNT(*), игнорируют неопределенные значения. -- Для замены неопределенных значений определенными используются функции NVL, NVL2 и COALESCE.

-- Сервер Oracle неявно сортирует данные в порядке возрастания, если используется предложение GROUP BY. Для того, чтобы изменить порядок сортировки, можно использовать опцию DESC после ORDER BY.

Использование функций AVG и SUM

Функции AVG, SUM, MIN, MAX применяются к столбцам, в которых можно хранить цифровые данные.

```
SELECT      AVG(salary), MAX(salary),
            MIN (salary), SUM(salary)
FROM        employees
WHERE       job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

В примере вычисляются средний, самый высокий, самый низкий оклад и сумма окладов всех торговых представителей.

Исключение групп: предложение HAVING

С помощью предложения HAVING их выходных данных исключаются некоторые группы.
Сервер Oracle обрабатывает предложение HAVING следующим образом:

1. Строки группируются.
2. К группе применяется групповая функция.
3. Выводятся группы, удовлетворяющие критериям в предложении HAVING.

Предложение HAVING может предшествовать предложению GROUP BY, но логичнее сделать предложение GROUP BY первым. Образование групп и вычисление групповых функций происходят до того, как к группам из списка SELECTприменяется предложение HAVING.

```
SELECT      [столбец,] групп_функция (столбец), …
FROM        таблица
[WHERE      условие]
[GROUP BY   выражение_группировки]
[HAVING     ограничивающее_условие]
[ORDER BY столбец];
```

Использование предложения HAVING

-- Предложение GROUP BY можно использовать без групповой функции в списке SELECT.
-- Для исключения строк после применения групповой функции требуются предложения GROUP BY и HAVING.

```
SELECT      department_id, MAX(salary)
FROM        employees
GROUP BY    department_id
HAVING   MAX(salary)>10000;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

В примере выводятся номера отделов и максимальный оклад только тех отделов, где он превышает 10000$;

Соединение таблиц

Пример: Чтобы вывести фамилию, номер отдела и местоположение отдела для служащего Matos, требуется дополнительное условие в предложении WHERE.

```
SELECT      last_name, employees.department_id, department_name
FROM        employess, departments
WHERE       employees.departmnet_id=departments.department_id
AND         last_name='Matos';
```

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping

Использование псевдонимов таблиц

```
SELECT      e.employee_id, e.last_name, e.department_id,
        d.department_id, d.location_id
FROM        employees e, departmnets d
WHERE       e.departmnet_id=d.departmnet_id;
```

-- Псевдонимы таблиц дают альтернативное имя таблице, уменьшают объем кода SQL и, следовательно, экономят память.

-- Псевдоним таблиц могут быть длиной до 30 символов;

-- Если в предложении FROM для указания таблицы используется псевдоним, этот псевдоним должен использоваться вместо имени таблицы во всем предложении SELECT;

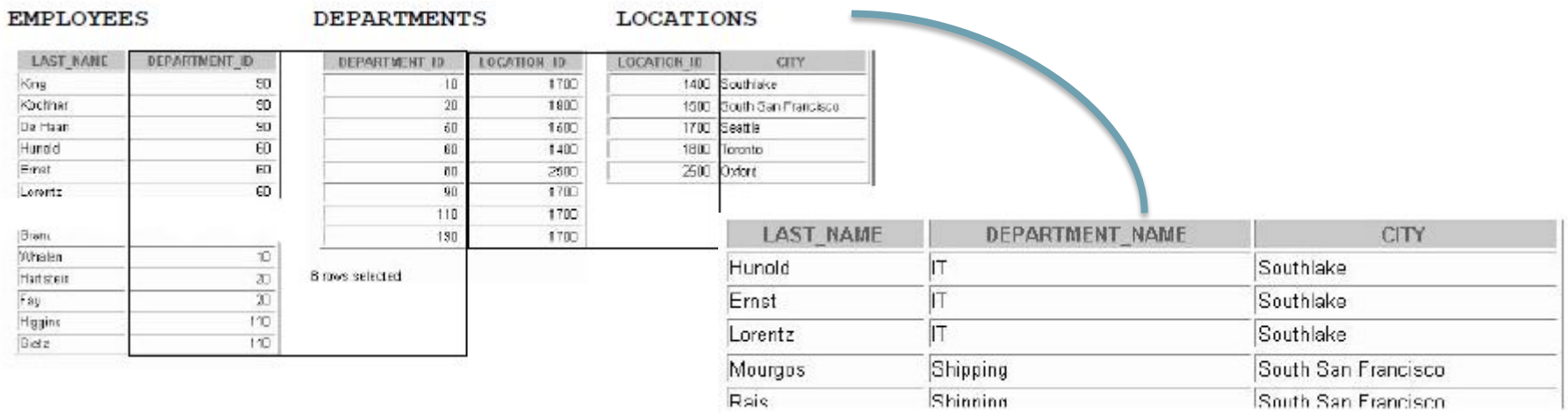
-- Следует выбирать осмысленные псевдонимы;

-- Действие псевдонима распространяется лишь на текущую команду SELECT.

Соединение более двух таблиц

Для соединения n таблиц требуется, по крайней мере, [n-1] условий соединения

```
SELECT      e.last_name, d.departmnet_name, l.city
FROM        employees e, departments d, locations l
WHERE       e.department_id = d.department_id
AND         d.location_id = l.location_id;
```



EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Bram	
Whalen	10
Hartstein	20
Fay	20
Higgins	110
Gietz	110

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

LAST_NAME	DEPARTMENT_NAME	CITY
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
Rajs	Shipping	South San Francisco

Синтаксис подзапросов

```
SELECT      список_выбора
FROM        таблица
WHERE       выражение оператор
            (SELECT     список_выбора
             FROM       таблица);
```

-- Подзапрос (внутренний запрос) выполняется один раз до главного запроса.
-- Результат подзапроса используется главным запросом (внешним запросом).

-- Подзапрос можно использовать в таких предложениях языка SQL как WHERE, HAVING, FROM

Многострочные подзапросы

Подзапросы возвращающие более одной строки называются ***многострочными.***

Многострочные подзапросы используют многострочные операторы сравнения.

Оператор	Значение
IN	***Равно любому члену списка***
ANY	***Сравнение значения с любым значением, возвращаемым подзапросом***
ALL	***Сравнение значения с каждым значением, возвращаемым подзапросом***

Пример:

```
SELECT      last_name, salary, department_id
FROM        employees
WHERE       salary IN (SELECT       MIN(salary)
                       FROM         employees
                       GROUP BY     department_id);
```

Успехов в освоении курса!

Савченко Наталья Александровна
savchenko.n@gubkin.ru