Лабораторная работа 2

Типы данных и встроенные функции

Каждый столбец таблицы реляционной базы данных должен содержать данные только одного типа. Тип данных определяет значения, которые могут быть присвоены элементам данного столбца и операции, в которых могут участвовать элементы данного столбца.

PostgreSQL имеет широкий набор встроенных типов, также, пользователи могут создавать и использовать собственные типы.

Числовые типы

Группа числовых типов включает:

- целочисленные типы;
- вещественные типы формата с фиксированной точкой;
- вещественные типы формата с плавающей точкой;
- последовательные типы.

Целочисленными типами являются: smallint, integer, bigint. Данные этих типов могут принимать только целочисленные значения, которые должны входить в заданный диапазон.

К вещественным типам формата с фиксированной точкой относятся: numeric(n,m) и decimal(n,m). Эти типы являются абсолютно идентичными по своим характеристикам. Они имеют два параметра: n – общее число десятичных разрядов в записи числа и m – число десятичных разрядов справа от десятичной точки. Данные этого типа обладают высокой точностью, но операции с такими данными выполняются медленнее по сравнению с другими числовыми типами.

Вещественными типами формата с фиксированной точкой являются: real и double precision. Данные этого типа могут употребляться при работе с данными обозначающими значение веса, времени и.т.д.

Характеристики числовых типов: размер и диапазон значений, приведены в таб.2.1.

Таблица 2.1. Характеристики числовых типов данных

Имя	**Размер**	**Диапазон значений**
smallint	2 байта	от -32768 до +32767
integer	4 байта	от -2147483648 до +2147483647
bigint	8 байт	от -9223372036854775808 до +9223372036854775807
numeric(n,m)	переменный	до 131072 цифр до десятичной точки; до 16383 цифр после десятичной точки
decimal(n,m)	переменный	до 131072 цифр до десятичной точки; до 16383 цифр после десятичной точки
real	4 байта	от 1E-37 до 1E+37 с точностью до 6 десятичных разрядов
double precision	8 байт	от 1E-307 до 1E+308 с точностью до 15 десятичных разрядов
smallserial	2 байта	от 1 до 32767
serial	4 байта	от 1 до 2147483647
bigserial	8 байт	от 1 до 9223372036854775807

В таблице 2.2. приведены основные функции, которые можно для работы с данными числовых типов

Таблица 2.2. Основные функции для работы с данными числовых типов

Функция	Описание
ROUND(x,n)	Выполняет округление числа x до ближайшего числа с заданной точностью n
TRUNC(x, n)	Усекает (отбрасывает) значащие цифры числа x справа без округления, с заданной точностью n
DIV(n,m)	Целая часть результата при делении n на m

MOD(n,m)	Возвращает остаток от деления n на m
POWER(x,n)	Возводит число x в степень n
SQRT(x)	Возвращает квадратный корень от числа x
EXP(n)	Возвращает значение экспоненты (Результат возведения e=2,718281 в степень n)
LN(n)	Вычисляет натуральный логарифм от числа n
LOG(n,m)	Производит вычисление логарифма числа n по основанию m
FACTORIAL(n)	Факториал числа n
RANDOM( )	Возвращает случайное значение в диапазоне 0.0 <= x < 1.0
SETSEED(Y)	Задает начальное значение для последующих `random()` вызовов; аргумент y должен быть в диапазоне от -1.0 до 1.0 включительно
GENERATE_SERIES (start,stop[,step])	Генерирует ряд значений от start до stop с шагом, равным step. Step по умолчанию равен 1, start, stop, step могут иметь тип integer или numeric

Рассмотрим примеры использования этих функций.

Запрос 2.1. Пример использования функции ROUND

```
SELECT ROUND(246.67), ROUND(246.67,1), ROUND(246.67,-1)
```

```
round|round|round|
-----+-----+-----+
  247|246.7|  250|
```

Запрос 2.2. Вывести значение зарплаты сотрудников из отдела 60, округленные до 1000.

```
SELECT employee_id, first_name, last_name, department_id, salary,
ROUND(salary,-3)
FROM  Employees
WHERE department_id=60
```

```
employee_id|first_name|last_name|department_id|salary |round|
-----------+----------+---------+-------------+-------+-----+
        103|Alexander |Hunold   |           60|9000.00| 9000|
        104|Bruce     |Ernst    |           60|6000.00| 6000|
        105|David     |Austin   |           60|4800.00| 5000|
        106|Valli     |Pataballa|           60|4800.00| 5000|
        107|Diana     |Lorentz  |           60|4200.00| 4000|
```

Запрос 2.3. Пример использования функции TRUNC

```
SELECT TRUNC(246.67), TRUNC(246.67,1), TRUNC(246.67,-1)
```

```
trunc|trunc|trunc|
-----+-----+-----+
  246|246.6|  240|
```

Запрос 2.4. Пример использования функции DIV

```
SELECT DIV(5,2), DIV(6.5,1),DIV (6.5,2.1)
```

```
div|div|div|
---+---+---+
  2|  6|  3|
```

Запрос 2.5. Пример использования функции MOD

```
SELECT mod(5,2), mod(6.5,1),mod(6.5,2.1)
```

```
mod|mod|mod|
---+---+---+
  1|0.5|0.2|
```

Запрос 2.6. Пример использования функции POWER

```
SELECT POWER(2,2),POWER(9,0.5),POWER(10,-1)
```

```
power|power              |power|
-----+-------------------+-----+
  4.0|3.0000000000000000|  0.1|
```

Вместо функции POWER можно использовать операцию возведения в степень x^a.

Запрос 2.7. Пример использования операции возведения в степень

```
SELECT 2^2, 9^0.5, 10^(-1)
```

```
?column?|?column?          |?column?|
--------+-------------------+--------+
     4.0|3.0000000000000000|     0.1|
```

Запрос 2.8. Пример использования функции EXP(),LN(),FACTORIAL()

```
SELECT EXP(2.0),LN(7.38906),FACTORIAL(4)
```

```
exp                |ln                  |factorial|
-------------------+--------------------+---------+
7.3890560989306502|2.0000005279521860|       24|
```

В PostgreSQL можно использовать функцию GENERATE_SERIES(), которая генерирует ряд значений между заданными начальной и конечной точками. Это может быть последовательность чисел или последовательность временных значений. В запросе 2.9 эта функция совместно с функцией RANDOM() используется для моделирования подбрасываний игрального кубика.

Запрос 2.9. Генерация последовательности случайных чисел с равномерным распределением, имеющих значения от 1 до 6.

```
SELECT s.i, TRUNC(RANDOM()*6)+1 as rnd
FROM GENERATE_SERIES(1,6) s(i);
```

```
i|rnd|
-+---+
1|5.0|
2|1.0|
3|1.0|
4|5.0|
5|6.0|
6|5.0|
```

Символьные типы

Символьными типами являются:

- строка фиксированной длинны;
- строка переменной длинны;
- строка неограниченной длинны.

Тип – строка фиксированной длинны, имеет обозначение **character(n)**, где n – максимальное число символов, которое может содержать строка. Для обозначения этого типа обычно используется псевдоним **char(n).**

Если присваиваемое значение будет короче заявленной длины n, то остальные разряды будут заполнены пробелами. Добавленные пробелы являются семантически незначимыми и не учитываются при сравнении двух значений имеющих тип character(n).

Попытка присвоить значение, которое будет длиннее заявленной длины n, приведет к возникновению ошибки.

Тип – строка переменной длинны, имеет обозначение **character varying(n)**, где n – максимальное число символов, которое может содержать строка. Для обозначения этого типа обычно используется псевдоним **varchar(n).**

Отличием от предыдущего типа является то, что при присвоении значения, которое будет короче заявленной длины n, дополнение пробелами не производится.

В PostgreSQL можно использовать тип: строка неограниченной длинны, который имеет обозначение **text**. Этого типа нет в стандарте SQL, но он используется во многих современных СУБД.

Конечные пробелы являются семантически значимыми в значениях имеющих типы **varchar(n)** и **text**.

Для работы с данными, имеющими строковые типы, можно использовать большое количество встроенных функций, некоторые из них приведены в таблице 2.3.

Таблица 2.3. Основные функции для работы с данными символьных типов

Функция	**Описание**
CONCAT(str1, str2…stri)	Выполняет конкатенацию строк str1, str2…stri
UPPER (str)	Осуществляет преобразование строки str в верхний регистр.
LOWER (str)	Осуществляет преобразование строки str в нижний регистр
INITCAP (str)	Осуществляет преобразование начальных букв каждого слова в верхний регистр.
LPAD(str, n [, char])	Возвращает строку str, дополненную слева символом char, до достижения строкой длины в n символов. Если char отсутствует, то добавляет пробелы

RPAD(str, n [, char])	Возвращает строку str, дополненную справа символом char, до достижения строкой длины в n символов. Если char отсутствует, то добавляет пробелы
LTRIM(str [, set])	Удаляет все символы с начала строки до первого символа, которого нет в наборе символов set. Если set отсутствует, то удаляет пробелы.
RTRIM(str [, set])	Удаляет символы, начиная от конца строки до первого символа, которого нет в наборе символов set. Если set отсутствует, то удаляет пробелы.
LENGTH(str)	Возвращает длину строки str в символах
REPLACE(str, search_str [, replace_str])	Осуществляет поиск образца search_str в строке str и каждое найденное вхождение заменяет на replace_str
SUBSTRING (str[FROM n] [FOR m])	Возвращает фрагмент строки str, начиная с символа n длиной m
STRPOS(str, search_str)	Возвращает позицию первого вхождения строки search_str в строку str
CHR(n)	Возвращает символ по его коду

Запрос 2.10. Вывести название товара, используя различные функции преобразования регистра.

```
SELECT
UPPER(Product_name) As UPPER,
LOWER(Product_name) As LOWER,
INITCAP(Product_name)As INITCAP
FROM Products
WHERE product_id = 50
```

```
upper                |lower                |initcap              |
---------------------+---------------------+---------------------+
MSI 24" OPTIX MAG241C|msi 24" optix mag241c|Msi 24" Optix Mag241c|
```

Функции LPAD() и RPAD() можно использовать для отображения результата выполнения запроса в виде, который боле удобен для восприятия.

Запрос 2.11 Вывод данных о зарплате сотрудников, зарплата которых больше 12000.

```
SELECT first_name||'  '||last_name || ' has a monthly salary of
' || salary || ' dollars.' AS Pay
FROM Employees
WHERE salary>12000
```

```
pay                                                              |
-----------------------------------------------------------------+
Steven  King has a monthly salary of 24000.00 dollars.           |
Neena  Kochhar has a monthly salary of 17000.00 dollars.         |
Lex  De Haan has a monthly salary of 17000.00 dollars.           |
John  Russell has a monthly salary of 14000.00 dollars.          |
Karen  Partners has a monthly salary of 13500.00 dollars.        |
Michael  Hartstein has a monthly salary of 13000.00 dollars.     |
```

Запрос 2.12. Вывод данных о зарплате сотрудников, зарплата которых больше 12000, с использованием функций LPAD() и RPAD().

```
SELECT RPAD(first_name||' '||last_name,20) ||
' has a monthly salary of ' ||
LPAD(TO_CHAR(salary,'99999D99'),9) || ' dollars.' AS Pay
FROM Employees
WHERE salary>12000;
```

```
pay                                                              |
-----------------------------------------------------------------+
Steven King          has a monthly salary of  24000,00 dollars.  |
Neena Kochhar        has a monthly salary of  17000,00 dollars.  |
Lex De Haan          has a monthly salary of  17000,00 dollars.  |
John Russell         has a monthly salary of  14000,00 dollars.  |
Karen Partners       has a monthly salary of  13500,00 dollars.  |
Michael Hartstein    has a monthly salary of  13000,00 dollars.  |
```

В этом примере следует обратить внимание на следующее:

- для того чтобы данные были выровнены по левому краю была использована функция RPAD();
- для выравнивания по правому краю использована функция LPAD().

Рассмотрим более подробно функцию **STRPOS(str, search_str),** которая часто используется при работе с символьными данными.

Функция **STRPOS(str, search_str)** возвращает номер позиции в строке str, начиная с которой строка **search_str** входит в строку **str.** Если вхождений не найдено, то функция возвращает значение 0.

Запрос 2.13. Использование функции STRPOS() для нахождения позиции первого пробела в названии товара.

```
SELECT product_name,strpos(product_name,' ')
FROM Products;
```

```
product_name                                                |strpos|
------------------------------------------------------------+------+
Corsair K70 RGB MK.2 Cherry MX Red (CH-9109010-RU)          |     8|
Logitech G810 Orion Spectrum (920-007750)                   |     9|
Logitech G910 Orion Spectrum RGB (920-008019)               |     9|
Samsung 24" S24F350FHI                                      |     8|
G.Skill TridentZ RGB                                        |     8|
Samsung 32" C32JG50QQI                                      |     8|
```

Типы даты и времени

PostgreSQL поддерживает все типы данных для представления даты и времени, предусмотренные стандартом SQL. Характеристики этих типов приведены в таб.2.4.

При вводе, значения этих типов должны быть заключены в кавычки и формально представляют собой строковые значения. В процессе выполнения операторов SQL эти значения приводятся к формату даты и времени.

Таблица 2.4. Характеристики типов даты и времени

Имя	Размер	Описание	Диапазон значений
DATE	4 байта	дата без указания времени	от 4713 до н.э. до 5874897 н.э.
TIME [(**p**)]	8 байт	время суток без указания часового пояса, с точностью p после точки в секундах	от 00.00.00.000000 до 24.00.00.000000
TIME [(**p**)]	8 байт	время суток с указанием	от 00.00.00.000000 +1559 до

Имя	Размер	Описание	Диапазон значений
WITH TIME ZONE		часового пояса, с точностью p после точки в секундах	24.00.00.000000-1559
TIMESTAMP [(**p**)]	8 байт	дата и время суток без указания часового пояса, с точностью p после точки в секундах	от 4713 до н.э. до 294276 н.э.
TIMESTAMP [(**p**)] WITH TIME ZONE	8 байт	дата и время суток с указанием часового пояса, с точностью p после точки в секундах	от 4713 до н.э. до 294276 н.э.
TSTZRANGE	16 байт	диапазон дат с подтипом timestamp with time zone	от 4713 до н.э. до 294276 н.э
INTERVAL [FIELDS][(**p**)]	16 байт	временной интервал	от -178000000 лет до +178000000 лет

Для работы с данными, имеющими тип даты и времени, можно использовать большое количество встроенных функций. В таблице 2.5. приведены основные функции, которые можно использовать при работе с данными этих типов.

Таблица 2.5. Основные функции для работы с данными типов даты и времени

Функция	Описание и пример
AGE(X,Y)	возвращает разницу между датами (X-Y), в виде интервала в годах, месяцах, днях, часах и т.д. AGE('2023-03-08', '2011-06-14') → 11 years 8 mons 24 days
AGE([timestamp]X)	возвращает разницу между текущей датой и датой X, в виде интервала в годах, месяцах, днях, часах и т.д. AGE(timestamp '2023-03-08') →1 mon 17 days
CURRENT_DATE	возвращает текущую дату CURRENT_DATE → 25-04-2023
CURRENT_TIME	возвращает текущее время суток с указанием часового пояса CURRENT_TIME → 13:43:14 +0300
CURRENT_TIMESTAMP	возвращает текущую дату и время с указанием часового пояса

	CURRENT_TIMESTAMP → 25-04-2023 13:49:30.057 +0300
NOW( )	возвращает текущую дату и время с указанием часового пояса NOW() → 26-04-2023 14:03:04.695 +0300
DATE_TRUNC(M,X)	обрезает значение X до заданной точности M (second; minute; hour; day; dow; month;year) SELECT CURRENT_DATE, DATE_TRUNC('MONTH', CURRENT_DATE)→ 25-04-2023|01-04-2023 00:00:00.000 +0300
EXTRACT(M FROM X)	извлекает заданную часть M (second; minute; hour; day; dow; month;year) из значения X SELECT CURRENT_DATE, EXTRACT('MONTH' FROM CURRENT_DATE) → 25-04-2023| 4|
JUSTIFY_INTERVAL(INTERVAL)	Преобразует значение INTERVAL в корректный формат даты и времени TIMESTAMP SELECT JUSTIFY_INTERVAL(INTERVAL'5000 hour 15 minute') → 6 mons 28 days 08:15:00

Стандарт ISO 8601 рекомендует использовать для вывода данных типа DATE формат 'yyyy-mm-dd' (год-месяц-день).

Значения, имеющие этот тип, могут участвовать в арифметических операциях, с некоторыми ограничениями. Например, разница меду двумя датами равна количеству дней прошедших между этими датами, но нельзя непосредственно складывать значения имеющие тип Date.

Прибавление целого значения n к значению типа Date эквивалентно прибавлению n дней к дате. Если в выражении участвует строка, содержащая значение даты или времени, то ее рекомендуется преобразовать к значению соответствующего типа, используя операцию приведения типа. Эта операция осуществляется путем размещения символа двойного двоеточия (::) и имени типа, к которому нужно привести строковое значение, например '05-4-2023'::date.

Рассмотрим примеры, в которых значения имеющие тип даты и времени участвуют в арифметических выражениях.

Запрос 2.14. Вывод заданного значения даты, увеличенного на 45 дней.

```
SELECT '05-4-2023'::DATE + 45 as nev_date
```

```
nev_date  |
----------+
20-05-2023|
```

Запрос 2.15. Вывод значения текущего даты и времени, увеличенного на 1 час и 10 минут

```
SELECT CURRENT_TIME, CURRENT_TIME + INTERVAL '1 hour 10 minute' AS
new_time
```

```
current_time  |new_time      |
--------------+--------------+
16:06:04 +0300|17:16:04 +0300|
```

В этом запросе содержится значение, имеющее тип **INTERVAL**. Рассмотрим синтаксис и правила использования этого типа данных.

Тип INTERVAL

Синтаксис:

```
INTERVAL [FIELDS][( p)]
```

где:

FIELDS – значение интервала;

p – точность после точки в секундах.

Значение интервала **FIELDS** представляет собой выражение

```
quantity unit [quantity unit]……..
```

где:

quantity – количество, которое может быть как положительным так и отрицательным;

unit – единица измерения: year, month, day, hour, minute, second, microsecond и некоторые другие единицы.

Рассмотрим примеры использования этого типа данных

Запрос 2.16.. Вывести значение даты заданной в виде интервала

```
SELECT  INTERVAL'10 day 4 month 2 year' AS new_date;
```

```
new_date              |
----------------------+
2 years 4 mons 10 days|
```

Запрос 2.17. Вычесть из текущей даты значение, заданное в виде интервала

```
SELECT (CURRENT_DATE - INTERVAL'10 day 4 month 2 year') AS
new_date;
```

```
new_date               |
-----------------------+
16-12-2020 00:00:00.000|
```

Запрос 2.18. Умножить значение времени, заданное в виде интервала

```
SELECT 10*INTERVAL'5 hour 15 minute' as new_time
```

```
new_time|
--------+
52:30:00|
```

В этом примере следует обратить внимание на то, что полученный результат не соответствует стандартному формату даты-времени. Для преобразования к такому формату следует использовать функцию **JUSTIFY_INTERVAL.**

Запрос 2.19. Умножить значение времени, заданное в виде интервала, и полученный результат преобразовать к стандартному формату даты-времени.

```
SELECT JUSTIFY_INTERVAL(INTERVAL'5000 hour 15 minute') as new_date
```

```
new_date               |
-----------------------+
6 mons 28 days 08:15:00|
```

Запрос 2.20. Для сотрудника employee_id=145 вывести период времени между датой приема на работу и сегодняшним днем.

```
SELECT AGE(hire_date)
FROM Employees WHERE employee_id=145;
```

```
age                     |
------------------------+
26 years 6 mons 25 days|
```

Используя функцию EXTRACT() можно извлечь определенное поле из значения, имеющего тип INTERVAL

Запрос 2.21. Вывести количество полных лет, которые проработал сотрудник 145.

```
SELECT EXTRACT(year FROM AGE(hire_date)) as "YEAR"
FROM Employees WHERE employee_id=145
```

```
YEAR|
----+
  26|
```

Функции преобразования типов данных

Эти функции предназначены для преобразования различных типов данных из одного формата в другой.

Преобразование чисел в строку символов

Числа, хранящиеся в базе данных, не имеют форматирования. Это означает, что они не имеют символов валюты, запятых, десятичных знаков и других параметров форматирования. Чтобы добавить форматирование, необходимо преобразовать число в строку символов. Для этого используется функция:

TO_CHAR(X,M)

Где:

X – число;

M – маска преобразования.

Маска преобразования может содержать элементы формата, представленные в таблице 2.8.

Таблица 2.8. Элементы числового форматирования

Элемент	Описание
9	Числовая позиция. Число 9-ок определяет ширину вывода.
0	Вывод начальных нулей до заданной ширины
. (D)	Десятичная точка (запятая)
, или G	Разделитель групп (тысяч)
L	Символ валюты (использует locale)

Если количество символов 9 в целой части шаблона будет меньше числа цифр целой части числа, то результат преобразования будет выведен в виде ###.##.

Ели количество символов 9 в дробной части шаблона будет меньше числа цифр дробной части числа, то будет выполнено округление.

Запрос 2.23. Примеры преобразования числа в строку символов

```
SELECT TO_CHAR(1475.29, '9999.99') As "9999.99",
       TO_CHAR(1475.29, '999.99') As "999.99",
       TO_CHAR(1475.29, '9999.9') As "9999.9",
       TO_CHAR(1475.29, '9999D99') As "9999D99",
       TO_CHAR(1475.29, '099999.90') As "099999.99",
       TO_CHAR(1475.29, '9,999.99') As "9,999.99",
       TO_CHAR(1475.29, 'L9,999.99') As "L9,999.99"
```

```
9999.99 |999.99 |9999.9 |9999D99 |099999.99 |9,999.99 |L9,999.99 |
--------+-------+-------+--------+----------+---------+----------+
 1475.29| ###.##| 1475.3| 1475,29| 001475.29| 1,475.29|$ 1,475.29|
```

Преобразование типов даты и времени в строку символов

Это преобразование выполняется для того чтобы отобразить значение имеющее тип даты и времени в требуемом виде. Для осуществления этого преобразования используется функция:

```
TO_CHAR(X,M)
```

Где:

X – значение имеющее тип даты и времени;

M – маска преобразования.

В таблице 2.9 приведены наиболее часто используемые элементы форматирования, которые может содержать маска преобразования.

Таблица 2.9. Основные элементы форматирования даты и времени

Элемент	Описание
YYYY	Все четыре цифры года
YY	Две последние цифры года
MM	Двузначный номер месяца
MONTH	Полное текстовое название месяца
MON	Первые три буквы названия месяца в верхнем регистре
D	Номер дня недели с воскресенья(1) по субботу (7)
ID	Номер дня недели с понедельника (1) по воскресенье (7)
DD	Двузначный номер дня месяца
DDD	Трехзначный номер дня года
DAY	Полное название дня недели SATURDAY
DY	Первые три буквы названия дня недели SAT
HH	Часы (01-12)
TZH	часы часового пояса
HH24	Часы (00-23)

MI	Минуты
SS	Секунды
MS	Миллисекунды

Значения, представляющие собой текст (MONTH, DAY), можно задавать в различных регистрах: прописной, строчный и.т.д...

Запрос 2.24. Примеры преобразования даты и времени в строку символов

```
SELECT TO_CHAR(CURRENT_TIMESTAMP, 'DD-MM-YYYY  HH24-MI-SS')
               as "DD-MM-YYYY  HH24-MI-SS",
       TO_CHAR(CURRENT_TIMESTAMP, 'DD-MM-YYYY') as "DD-MM-YYYY",
       TO_CHAR(CURRENT_TIMESTAMP, 'HH24-MI-SS') as "HH24-MI-SS",
       TO_CHAR(CURRENT_TIMESTAMP, 'DAY') as "DAY",
       TO_CHAR(CURRENT_TIMESTAMP, 'TZH') as "TZH"
```

```
DD-MM-YYYY  HH24-MI-SS|DD-MM-YYYY|HH24-MI-SS|DAY      |TZH|
----------------------+----------+----------+---------+---+
07-05-2023  11-46-11  |07-05-2023|11-46-11  |SUNDAY   |+03|
```

Преобразование строки символов в число

Для преобразования символьного значения в число используется функция

TO_NUMBER(X, M})

Где:

X – символьное представление числа;
M – маска преобразования.

X может содержать цифры и символы, которые соответствуют заданному формату.

M определяет, как нужно интерпретировать символьное представление числа, может содержать те же элементы формата, которые были определены для функции TO_CHAR.

Преобразование строки символов к типам даты и времени

Для преобразования строки символов в значение имеющее тип даты и времени используются функции:

TO_DATE(X, M)

TO_TIMESTAMP(X, M)

X – содержит символьное значение даты и времени.

M – маска преобразования определяет как нужно интерпретировать символьное представление даты и времени.

Маска может содержать элементы формата, представленные в таблице 2.9 . При использовании функции TO_DATE() следует использовать только те элементы формата, которые соответствуют дате (день, месяц, год).

Запрос 2.25. Примеры использования функции TO_DATE

```
SELECT TO_DATE ('01-SEP-2018', 'DD-MON-YYYY')
       As "DD-MON-YYYY'",
       TO_DATE ('09/01/18', 'MM/DD/YY') As "MM/DD/YY",
       TO_DATE ('01092018', 'DDMMYYYY') As "DDMMYYYY"
```

```
DD-MON-YYYY'|MM/DD/YY  |DDMMYYYY  |
------------+----------+----------+
  01-09-2018|01-09-2018|01-09-2018|
```

В этом примере строка преобразуется в дату, а дата выводится в установленном формате даты.

Преобразование значений NULL

Если при вводе новой строки в таблицу столбцу не будет присвоено значение, то этот столбец будет иметь значение NULL – неопределенно. Это может происходить по двум основным причинам.

Первая причина: в момент ввода строки значение столбца неизвестно, в этом случае, значение будет присвоено позже.

Вторая причина: значение не может быть присвоено исходя из правил предметной области. Для рассматриваемой базы данных, вторую причину можно пояснить на примере столбца commission_pct таблицы Employees. Некоторым сотрудникам полагаются комиссионные, столбец commission_pct содержит значение комиссионных. Зарплата таких сотрудников рассчитывается по формуле: Salary*(1+ commission_pct). У сотрудников, которым комиссионные не полагаются, значение столбца commission_pct не может быть определено.

При работе с арифметическими выражениями следует иметь в виду следующее: арифметическое выражения вернет значение NULL, если один или несколько операндов будет иметь значение NULL. Результатом операции сравнения будет NULL, если один или оба операнда будут иметь значение NULL.

Для корректной обработки данных, которые могут иметь значения NULL, следует использовать функцию COALESCE().

Функция COALESCE

Используется в выражениях, элементы которых могут иметь значение NULL, и имеет следующий синтаксис:

```
COALESCE (X1,X2,…XN)
```

Функция возвращает первое не NULL значение. Если все ее аргументы равны NULL, то функция возвращает NULL.

Запрос 2.26. Вывести данные о полной зарплате сотрудников, которые работают в отделе 30. Значение полной зарплаты равно salary*(1+commission_pct).

```
SELECT employee_id, first_name, last_name, salary,
commission_pct as com_pct,
ROUND(COALESCE(salary*(1+commission_pct),salary)) AS
total_salary
FROM  Employees
WHERE department_id =30
ORDER BY  total_salary DESC
```

employee_id	first_name	last_name	salary	com_pct	total_salary
114	Den	Raphaely	11000.00	0.200	13200
116	Shelli	Baida	2900.00	0.300	3770
115	Alexander	Khoo	3100.00		3100
117	Sigal	Tobias	2800.00	0.100	3080
118	Guy	Himuro	2600.00		2600
119	Karen	Colmenares	2500.00		2500

Без использования функции COALESCE() полная зарплата сотрудников, у которых commission_pct имеет значение NULL, также имела бы значение NULL.

Задание

Задача 1. Вывести данные о сотрудниках из отдела 60, имеющих нечетный рейтинг.

Задача 2. Для сотрудников, зарплата которых больше 10000, выведите столбец, который должен содержать полное имя сотрудника, зарплату и несколько звездочек *, по одной звездочке на каждую 1000 зарплаты.

Задача 3. Вывести данные о товарах, название которых содержит слово AMD и не содержит слово RYZEN. Предусмотреть то, что эти слова, в названии товара, могут быть представлены в разных регистрах.

Задача 4. Выведите названия отделов, которые состоят более чем из одного слова.

Задача 5. Вывести данные о сотрудниках, которые были приняты на работу 21 апреля.

Задача 6. Вывести данные о договорах, которые были оформлены в воскресенье.

Задача 7. Вывести данные о размере премии сотрудников, которые работают в отделе 30. Размер премии равен: зарплате с учетом комиссионных, если сотрудник получает комиссионные, и равен зарплате умноженной на 1.2, если сотрудник не получает комиссионные.