2.3. Преобразование вывода и встроенные функции

В SQL реализованы операторы преобразования данных и встроенные функции, предназначенные для работы со значениями столбцов и/или константами в выражениях. Использование этих операторов допустимо в запросах везде, где можно использовать выражения.

2.3.1. Числовые, символьные и строковые константы

Несмотря на то, что SQL работает с данными в понятиях строк и столбцов таблиц, имеется возможность применения значений выражений, построенных с использованием встроенных функций, констант, имен столбцов, которые определяются как своего рода виртуальные столбцы. Они помещаются в списке столбцов и могут сопровождаться псевдонимами.

Если в запросе вместо спецификации столбца SQL обнаруживает *число*, то оно интерпретируется как *числовая константа*.

Символьные константы должны указываться в одинарных кавычках. Если одинарная кавычка должна выводиться как часть строковой константы, то ее нужно предварить другой одинарной кавычкой

Например, результатом выполнения запроса

```
SELECT 'Фамилия', SURNAME, 'Имя', NAME, 100
    FROM STUDENT;
```

является таблица следующего вида

	SURNAME		NAME	
Фамилия	Иванов	Имя	Иван	100
Фамилия	Петров	Имя	Петр	100
Фамилия	Сидоров	Имя	Вадим	100
Фамилия	Кузнецов	Имя	Борис	100
Фамилия	Зайцева	Имя	Ольга	100
Фамилия	Павлов	Имя	Андрей	100
Фамилия	Котов	Имя	Павел	100
Фамилия	Лукин	Имя	Артем	100
Фамилия	Петров	Имя	Антон	100
Фамилия	Белкин	Имя	Вадим	100
……	……	……	……	……

2.3.2. Арифметические операции для преобразования числовых данных

- Унарный (одиночный) оператор “**–**” (знак минус) изменяет знак числового значения, перед которым он стоит, на противоположный.
- Бинарные операторы “**+**”, “**–**“, “*****” и “**/**” предоставляют возможность выполнения арифметических операций сложения, вычитания, умножения и деления.

Например, результат запроса

```
SELECT SURNAME, NAME, STIPEND, –(STIPEND*KURS)/2
    FROM STUDENT
    WHERE KURS = 4 AND STIPEND > 0;
```

будет выглядеть следующим образом

SURNAME	NAME	STIPEND	KURS	
Сидоров	Вадим	150	4	-300
Петров	Антон	200	4	-400
……	……	……	……	……

2.3.3. Символьная операция конкатенации строк

Операция конкатенации “**||**” позволяет соединять (“склеивать”) значения двух или более столбцов символьного типа или символьных констант в одну строку.

Эта операция имеет синтаксис

<*значимое символьное выражение*> {**||**}<*значимое символьное выражение*>.

Например:

```
SELECT SURNAME || '_' || NAME, STIPEND
    FROM STUDENT
    WHERE KURS = 4 AND STIPEND > 0;
```

Результат запроса будет выглядеть следующим образом

	STIPEND
Сидоров_Вадим	150
Петров_Антон	200
……	……

2.3.4. Символьные функции преобразования букв различных слов в строке

- **LOWER** – перевод в строчные символы (нижний регистр)

  LOWER (<*строка*>)

- **UPPER** – перевод в прописные символы (верхний регистр)

  UPPER (<*строка*>)

- **INITCAP** – перевод первой буквы каждого слова строки в заглавную (прописную)

  INITCAP(<*строка*>)

  Например:

```
SELECT LOWER(SURNAME), UPPER(NAME)
    FROM STUDENT
    WHERE KURS = 4 AND STIPEND > 0;
```

  Результат запроса будет выглядеть следующим образом

SURNAME	NAME
Сидоров	ВАДИМ
Петров	АНТОН
……	……

2.3.5. Символьные строковые функции

- **LPAD** – дополнение строки слева

  LPAD (<*строка*>, <*длина*> [, <*подстрока*>])

  - <*строка*> дополняется **слева** указанной в <*подстроке*> последовательностью символов до указанной <*длины*> (возможно, с повторением последовательности);
  - если <*подстрока*> не указана, то по умолчанию <*строка*> дополняется пробелами;
  - если <*длина*> меньше длины <*строки*>, то исходная <*строка*> усекается слева до заданной <*длины*>.

- **RPAD** – дополнение строки справа

  RPAD (<*строка*>, <*длина*> [, <*подстрока*>])

  - <*строка*> дополняется **справа** указанной в <*подстроке*> последовательностью символов до указанной <*длины*> (возможно, с повторением последовательности);
  - если <*подстрока*> не указана, то по умолчанию <*строка*> дополняется пробелами;
  - если <*длина*> меньше длины <*строки*>, то исходная <*строка*> усекается справа до заданной <*длины*>.

- **LTRIM** – удаление левых граничных символов

  LTRIM (<*строка*> [,<*подстрока*>])

  - из <*строки*> удаляются слева символы, указанные в <*подстроке*>;
  - если <*подстрока*> не указана, то по умолчанию удаляются пробелы;
  - в <*строку*> справа добавляется столько пробелов, сколько символов слева было удалено, то есть длина <*строки*> остается неизменной.

- **RTRIM** – удаление правых граничных символов

  RTRIM (<*строка*> [,<*подстрока*>])

  - из <*строки*> удаляются справа символы, указанные в <*подстроке*>;
  - если <*подстрока*> не указана, то по умолчанию удаляются пробелы;
  - в <*строку*> слева добавляется столько пробелов, сколько символов справа было удалено, то есть длина <*строки*> остается неизменной.

Функции **LTRIM** и **RTRIM** рекомендуется использовать при написании условных выражений, в которых сравниваются текстовые строки. Дело в том, что наличие начальных или конечных пробелов в сравниваемых операндах может исказить результат сравнения.

Например, константы ‘ AAA’ и ‘ AAA ’ не равны друг другу.

- **SUBSTR** – выделение подстроки

  SUBSTR (<*строка*>,<*начало*> [,<*количество*>])

  - из <*строки*> выбирается заданное <*количество*> символов, начиная с указанной позиции в строке <*начало*>;

- если <*количество*> не задано, символы выбираются с <*начала*> и до конца <*строки*>.
- возвращается подстрока, содержащая число символов, заданное параметром <*количество*>, либо число символов от позиции, заданной параметром <*начало*> до конца *строки*;
- если указанное <*начало*> превосходит длину <*строки*>, то возвращается строка, состоящая из пробелов. Длина этой строки будет равна заданному <*количеству*> или исходной длине <*строки*> (при не заданном <*количестве*>).

- **INSTR** – поиск подстроки

  INSTR(<*строка*>,<*подстрока*> [,<*начало поиска*>
  [,<*номер вхождения*>]])

  - <*начало поиска*> задает начальную позицию в строке для поиска <*подстроки*>. Если не задано, то по умолчанию принимается значение 1;
  - <*номер вхождения*> задает порядковый номер искомой подстроки. Если не задан, то по умолчанию принимается значение 1;
  - значимые выражения в <*начале поиска*> или в <*номере вхождения*> должны иметь беззнаковый целый тип или приводиться к этому типу;
  - тип возвращаемого значения – **INT;**
  - функция возвращает позицию найденной подстроки.

- **LENGTH –** определение длины строки

  LENGTH(<*строка*>)

  - длина <*строки*>, тип возвращаемого значения – **INT**;
  - функция возвращает **NULL**, если <*строка*> имеет **NULL**-значение.

  Примеры запросов, использующих строковые функции.

  Результат запроса

```
SELECT LPAD (SURNAME, 10, '@'), RPAD (NAME, 10, '$')
    FROM STUDENT
    WHERE KURS = 3 AND STIPEND > 0;
```

будет выглядеть следующим образом

@@@@Петров	Петр$$$$$$
@@@@Павлов	Андрей$$$$
@@@@@Лукин	Артем$$$$$
……	……

А запрос

```
SELECT SUBSTR(NAME, 1, 1) || '.' || SURNAME, CITY, LENGTH(CITY)
    FROM STUDENT
    WHERE KURS IN(2, 3, 4) AND STIPEND > 0;
```

выдаст результат

	CITY	
П.Петров	Курск	5
С.Сидоров	Москва	6
О.Зайцева	Липецк	6
А.Лукин	Воронеж	7
А.Петров	**NULL**	**NULL**
……	……	……

2.3.6. Функции работы с числами

- **ABS** – абсолютное значение

  ABS(<*значимое числовое выражение*>)

- **FLOOR** – урезает значение числа с плавающей точкой до наибольшего целого, не превосходящего заданное число

  FLOOR(<*значимое числовое выражение*>)

- **CEIL** – самое малое целое, которое равно или больше заданного числа

  CEIL(<*значимое числовое выражение*>)

- Функция округления – **ROUND**

  ROUND(<*значимое числовое выражение*>,<*точность*>)

  аргумент <*точность*> задает точность округления (см. **пример** ниже)

- Функция усечения – **TRUNC**

  TRUNC(<*значимое числовое выражение*>,<*точность*>)

- Тригонометрические функции – **COS**, **SIN**, **TAN**

  COS(<*значимое числовое выражение*>)
  SIN(<*значимое числовое выражение*>)
  TAN(<*значимое числовое выражение*>)

- Гиперболические функции – **COSH, SINH, TANH**

  COSH(<*значимое числовое выражение*>)
  SINH(<*значимое числовое выражение*>)
  TANH(<*значимое числовое выражение*>)

- Экспоненциальная функция – **(EXP)**

  EXP(<*значимое числовое выражение*>)

- Логарифмические функции – **(LN**, **LOG)**

  LN(<*значимое числовое выражение*>)
  LOG(<*значимое числовое выражение*>)

- Функция возведения в степень – **POWER**

  POWER(<*значимое числовое выражение*>,<*экспонента*>)

- Определение знака числа – **SIGN**

  SIGN(<*значимое числовое выражение*>)

- Вычисление квадратного корня – **SQRT**

  SQRT(<*значимое числовое выражение*>)

  Пример.

  Запрос

```
SELECT UNIV_NAME, RATING, ROUND(RATING, -1), TRUNC(RATING, -1)
    FROM UNIVERSITY;
```

  Вернет результат

UNIV_NAME	RATING		
МГУ	606	610	600
ВГУ	296	300	290
НГУ	345	350	340
РГУ	416	420	410
БГУ	326	330	320
ТГУ	368	370	360
ВГМА	327	330	320
……	……	……	……

2.3.7. Функции преобразования значений

- Преобразование в символьную строку – **TO_CHAR**

  TO_CHAR(<*значимое выражение*>[,<*символьный формат*>])

  - <*значимое выражение*> должно представлять числовое значение или значение типа дата-время;
  - для числовых значений <*символьный формат*> должен иметь синтаксис [S]9[9…][.9[9…]], где S – представление знака числа (при отсутствии предполагается без отображения знака), 9 – представление цифр-знаков числового значения (для каждого знакоместа). Символьный формат определяет вид отображения чисел. По умолчанию для числовых значений используется формат '999999.99';
  - для значений типа **ДАТА-ВРЕМЯ** <*символьный формат*> имеет вид (то есть вид отображения значений даты и времени):
    – в части даты
    'DD-Mon-YY'
    'DD-Mon-YYYY'
    'MM/DD/YY'
    'MM/DD/YYYY'
    'DD.MM.YY'
    'DD.MM.YYYY'
    – в части времени
    'HH24'

‘HH24:MI’
‘HH24:MI:SS’
‘HH24:MI:SS.FF’

где:

HH24 - часы в диапазоне от 0 до 24
MI – минуты
SS – секунды
FF – тики (сотые доли секунды)

При выводе времени в качестве разделителя по умолчанию используется двоеточие (**:**), но при желании можно использовать любой другой символ.

Возвращаемое значение – символьное представление <*значимого выражения*> в соответствии с заданным <*символьным форматом*> преобразования.

- Преобразование из символьного значения в числовое – **TO_NUMBER**

  TO_NUMBER(<*значимое символьное выражение*>)

  При этом <*значимое символьное выражение*> должно задавать символьное значение числового типа.

- Преобразование символьной строки в дату – **TO_DATE**

  TO_DATE(<*значимое символьное выражение*> [,<*символьный формат*>])

  - <*значимое символьное выражение*> должно задавать символьное значение типа **ДАТА-ВРЕМЯ**.
  - <*символьный формат*> должен описывать представление значения типа **ДАТА-ВРЕМЯ** в <*значимом символьном выражении*>. Допустимые форматы (в том числе и формат по умолчанию) приведены выше.

Возвращаемое значение – <*значимое символьное выражение*> во внутреннем представлении. Тип возвращаемого значения – **DATE**. Операции над значениями типа **DATE**

Над значениями типа **DATE** разрешены следующие операции:

- бинарная операция сложения;
- бинарная операция вычитания.

В бинарных операциях один из операндов должен иметь значение отдельного элемента даты: только год, или только месяц, или только день.

Например:

при добавлении к дате ‘22.05.1998’ пяти лет получится дата ‘22.05.2003’;
при добавлении к этой же дате девяти месяцев получится дата ‘22.02.1998’;
при добавлении 10-ти дней получим ‘01.06.1998’.

При сложении двух полных дат, например, ‘22.05.1998’ и ‘01.12.2000’ результат непредсказуем.

Пример.

Запрос

```
SELECT SURNAME, NAME, BIRTHDAY,
                      TO_CHAR(BIRTHDAY, ‘DD-Mon-YYYY’),
                      TO_CHAR(BIRTHDAY, ‘DD.MM.YY’)
    FROM STUDENT;
```

Вернет результат

SURNAME	NAME	BIRTHDAY		
Иванов	Иван	3/12/1982	3-дек-1982	3.12.82
Петров	Петр	1/12/1980	1-дек-1980	1.12.80
Сидоров	Вадим	7/06/1979	7-июн-1979	7.06.79
Кузнецов	Борис	8/12/1981	8-дек-1981	8.12.81
Зайцева	Ольга	1/05/1981	1-май-1981	1.05.81
Павлов	Андрей	5/11/1979	5-ноя-1979	5.11.79
Котов	Павел	**NULL**	**NULL**	**NULL**
Лукин	Артем	1/12/1981	1-дек1981	1.12.81
Петров	Антон	5/08/1981	5-авг-1981	5.08.81
Белкин	Вадим	7/01/1980	7-янв-1980	7.01.80
……	……	……	……	……

Функция **CAST** является средством явного преобразования данных из одного типа в другой. Синтаксис этой команды имеет вид

CAST <*значимое выражение*> **AS** <*тип данных*>

- <*значимое выражение*> должно иметь числовой или символьный тип

языка SQL (возможно, с указанием длины, точности и масштаба) или быть **NULL**-значением.

- любое числовое выражение может быть явно преобразовано в любой другой числовой тип.
- символьное выражение может быть преобразовано в любой числовой тип. При этом в результате символьного выражения отсекаются начальные и конечные пробелы, а остальные символы преобразуются в числовое значение по правилам языка SQL.
- если явно заданная длина символьного типа недостаточна и преобразованное значение не размещается в нем, то результативное значение усекается справа.
- возможно явное преобразование символьного типа в символьный с другой длиной. Если длина результата больше длины аргумента, то значение дополняется пробелами; если меньше, то усекается.
- **NULL**-значение преобразуется в **NULL**-значение соответствующего типа.
- числовое выражение может быть преобразовано в символьный тип.

Пример.

```
SELECT CAST STUDENT_ID AS CHAR(10)
    FROM STUDENT;
```

УПРАЖНЕНИЯ

1. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала один столбец, содержащий последовательность разделенных символом “;” (точка с запятой) значений всех столбцов этой таблицы, и при этом текстовые значения должны отображаться прописными символами (верхний регистр), то есть быть представленными в следующем виде: 10;КУЗНЕЦОВ;БОРИС;0;БРЯНСК;8/12/1981;10.
2. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде: Б.КУЗНЕЦОВ; место жительства-БРЯНСК; родился - 8.12.81.
3. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде:

б.кузнецов; место жительства-брянск; родился: 8-дек-1981.

4. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде:
   Борис Кузнецов родился в 1981 году.
5. Вывести фамилии, имена студентов и величину получаемых ими стипендий, при этом значения стипендий должны быть увеличены в 100 раз.
6. То же, что и в задаче 4, но только для студентов 1, 2 и 4-го курсов и таким образом, чтобы фамилии и имена были выведены прописными буквами.
7. Составьте запрос для таблицы UNIVERSITY таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде:
   Код-10; ВГУ-г.ВОРОНЕЖ; Рейтинг=296.
8. Тоже, что и в задаче 7, но значения рейтинга требуется округлить до первого знака (например, значение 382 округляется до 400).

2.4. Агрегирование и групповые функции

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную) информацию, выполняя операции над группой строк таблицы. Для задания в **SELECT-**запросе агрегирующих операций используются следующие ключевые слова:

- **COUNT** определяет количество строк или значений поля, выбранных посредством запроса, и не являющихся **NULL**-значениями;
- **SUM** – вычисляет арифметическую сумму всех выбранных значений данного поля;
- **AVG** вычисляет среднее значение для всех выбранных значений данного поля;
- **MAX** вычисляет наибольшее из всех выбранных значений данного поля;
- **MIN** вычисляет наименьшее из всех выбранных значений данного поля.

В **SELECT-**запросе агрегирующие функции используются аналогично именам полей, при этом последние (имена полей) используются в качестве аргументов этих функций.

Функция **AVG** предназначена для подсчета среднего значения поля на множестве записей таблицы.