Оглавление

Выражения в SQL

В списке полей оператора select могут участвовать не только имена полей, но и более общие конструкции называемые выражениями.

Типы выражений

- NULL
- Числовые константы. Пример: 2, 3.14159216, 10.2E+21
- Литералы (строковые константы). Пример: ‘Это литерал’
- Литералы даты и времени. Пример: ‘20170420’
- Сцепление (конкатенация) строк
- Математические выражения
  - **+** сложение
  - **–** вычитание
  - ***** умножение
  - **/** деление
  - **%** деление по модулю
- Функции

```
select
	null as [NULL value]
	, 3.1415926 as [PI]
	, 'This is string literal' as [Literal]
	, '2017-04-20 9:43PM' as [DateTime literal]
	, 'First ' + ' second' as [Concatenation]
	, 2 * 2 as [Four]
	, getdate() as [Current date and time]
```

```
select
	null as 'NULL value'
	, 3.1415926 as 'PI'
	, 'This is string literal' as 'Literal'
	, '2017-04-20 9:43PM' as 'DateTime literal'
	, 'First ' + ' second' as 'Concatenation'
	, 2 * 2 as 'Four'
	, current_date() as 'Current date and time'
```

Оператор CASE

Используется для оценки списка условий и возвращает одно из нескольких возможных значений.

Выражение CASE имеет два формата:

- **простое выражение CASE.** Для определения результата сравнивает выражение с набором простых выражений;
- **поисковое выражение CASE**. Для определения результата вычисляет набор логических выражений.

Оба формата поддерживают дополнительный аргумент ELSE.

Выражение CASE можно использовать в таких инструкциях, как SELECT, UPDATE, DELETE и SET, а также в таких предложениях, как select_list, IN, WHERE, ORDER BY и HAVING.

Простое выражение CASE

```
CASE <выражение>
       WHEN <значение проверки> THEN <итоговое выражение>
       [WHEN <значение проверки> THEN <итоговое выражение>]
       [ELSE <итоговое значение>]
END
```

Поисковое предложение CASE

```
CASE
       WHEN <условие проверки> THEN <итоговое выражение>
       [WHEN <условие проверки> THEN <итоговое выражение>]
       [ELSE <итоговое значение>]
END
```

Пример простого выражения CASE

```
select
       surname as 'Фамилия'
       , name as 'Имя'
       , case gender
              when 'm' then 'Мужской'
              when 'f' then 'Женский'
    end as 'Пол'
from STUDENTS
```

Пример поискового выражения CASE

```
select
       surname as 'Фамилия'
       , name as 'Имя'
       , birthday as 'Дата рождения'
       , month(BIRTHDAY) as 'Месяц рождения'
       , case
              when month(BIRTHDAY) between 1 and 2 or month(BIRTHDAY) = 12 then 'Родился
зимой'
              when month(BIRTHDAY) between 3 and 5  then 'Родился весной'
              when month(BIRTHDAY) in (6,7,8)  then 'Родился летом'
              when month(BIRTHDAY) >= 9 and month(BIRTHDAY) <= 11
then 'Родился осенью'
    end as 'Сезон'
from STUDENTS
```

MySQL

```
select
       surname 'Фамилия'
       , name as 'Имя'
       , birthday 'Дата рождения'
       , month(BIRTHDAY) 'Месяц рождения'
       , case
              when month(BIRTHDAY) between 1 and 2
                     or month(BIRTHDAY) = 12
                     then 'Родился зимой'
              when month(BIRTHDAY) between 3 and 5  then 'Родился весной'
              when month(BIRTHDAY) in (6,7,8)  then 'Родился летом'
              when month(BIRTHDAY) >= 9 and month(BIRTHDAY) <= 11
       then 'Родился осенью'
    end as 'Сезон'
from STUDENTS
```

Пример использования CASE в операторах фильтра и сортировки

```
select
	*
from STUDENTS
where
	city like
		case
			when month(birthday) between 1 and 6 then 'K%'
			when month(birthday) between 7 and 12 then 'X%'
		end
order by
	case gender
		when 'f' then name
		when 'm' then SURNAME
	end;
```

Конкатенация (сцепление) строк

Для обозначения операции конкатенации строк разные сервера используют разные символы. MS SQL Server использует символ «+», а Oracle «||». Стандарт определяет функцию

CONCAT(str1, str2, ….strn)

```
select surname + ' ' + name  as fullname from STUDENTS
select concat(surname,' ', name)  as fullname from STUDENTS
```

Строковые функции T-SQL

Полный перечень строковых функций t-sql можно найти по ссылке:

String Functions (Transact-SQL)

Строковые функции MySQL

Строковые функции MySQL

Некоторые важные строковые функции

- **LOWER (character_expression)** - Возвращает символьное выражение после преобразования символов нижнего регистра в символы верхнего регистра.
- **UPPER (character_expression)** - Возвращает символьное выражение, в котором символы нижнего регистра преобразованы в символы верхнего регистра.
- **LEN (string_expression)** - Возвращает количество символов указанного строкового выражения, исключая конечные пробелы
- **REPLACE (string_expression , string_pattern , string_replacement)** - Заменяет все вхождения указанного строкового значения другим строковым значением.
- **SUBSTRING (expression ,start , length)** - Возвращает часть символьного или текстового в SQL Server.
- **LEFT (character_expression , integer_expression)** - Возвращает указанное число символов символьного выражения слева.
- **LTRIM (character_expression)** - Возвращает символьное выражение после удаления начальных пробелов.

Пример:

```
select
```

```
       concat(surname, ' ', LEFT(UPPER(name), 1), '.') as sname
from STUDENTS

select
       surname + ' ' +  LEFT(UPPER(name), 1) + '.' as sname
from STUDENTS

select replace('This xxx a string', 'xxx', 'is') as repex
```

Математические выражения

В SQL предложении могут участвовать математические операции:

- [**+**] : сложение
- [**-**] : вычитание
- [*****] : умножение
- [**/**] : деление
- [**%**] : деление по модулю (остаток от деления)

```
select
    SURNAME
     , NAME
     , BIRTHDAY
     , COURSE
     , CITY
from STUDENTS
where CITY = 'Киев' and COURSE % 2 = 1;
```

Математические функции

- **ABS (X)** - абсолютное значение
- **FLOOR (X)** - урезает значение числа с плавающей точкой до наибольшего целого, не превосходящего заданное число
- **CEILING (X)** - самое малое целое, равное или большее заданного числа
- **ROUND (X,<точность>)** - функция округления, аргумент <точность> задает точность округления
- **COS (X), SIN (X), TAN (X)** - тригонометрические функции
- **EXP (X)** - экспоненциальная функция
- **LN (X), LOG (X)** - логарифмические функции
- **POWER (X,<экспонента>)** - функция возведения в степень
- **SIGN (X)** - определение знака числа
- **SQRT (X)** - вычисление квадратного корня

где X - значимое числовое выражение

Полный перечень математических функций Mathematical Functions (Transact-SQL)
Полный перечень математических функций MySQL

Функции преобразования типов

Смешивание и сопоставление типов данных в рамках одной и той же операции неизбежно приводит к неявному преобразованию типов, в результате чего SQL преобразует один из операндов так, чтобы сделать его таким же, как и другие операторы.

Явные и неявные преобразования

Преобразование типов данных бывает явным и неявным.

Неявное преобразование скрыто от пользователя. SQL Server автоматически преобразует данные из одного типа в другой. Например, если **smallint** сравнивается с **int**, **smallint** неявно преобразуется в **int** до начала операции сравнения.

Явное преобразование выполняется с помощью функций CAST и CONVERT.

Неявное (Implicit) преобразование типов

Позволять SQL выполнять неявные преобразования типов не такая уж плохая практика, когда заранее известно какие значения данных будут получены, но иногда это может привести к неожиданным результатам! Например, добавление '1' к 1 не является проблемой, потому что строка '1' будет просто преобразована в числовой тип. С другой стороны, что-то вроде 'r' + 1 становится менее очевидным.

Таким образом, несмотря на то, что некоторые преобразования кажутся логичными и безвредными, есть много примеров, когда неявное преобразование может привести к некоторому очень неустойчивому поведению. Особенно проблематичные типы включают типы datetime, timestamp, data, а также смешанные числовые типы, такие как целые числа и числа с плавающей точкой. Есть случаи, когда SQL должен преобразовывать несколько значений, но не знает, к какому типу нужно сделать преобразование. В таких случаях, в лучшем случае, генерируется сообщение об ошибке. В противном случае может произойти преобразование в неверный тип, что может привести к runtime ошибке.

Явное (explicit) преобразование типов

По причинам, описанным выше, часто необходимо самостоятельно обрабатывать все преобразования, а не позволять SQL делать эту работу неявным, а иногда и неочевидным способом. Для этого предназначены функции CONVERT () и CAST (). Обе функции принимают выражение одного типа и дают результат другого типа.

CAST (expression AS data_type [(length)])
CONVERT (data_type [(length)] , expression [, style])

Для MySQL функция CONVERT существует две синтаксические формы функции CONVERT: одна предназначена для конвертирования типов и имеет вид:

CONVERT (expression, data_type [(length)]) ,

а вторая предназначена для конвертирования кодировок символов (character set) и имеет вид

CONVERT(value USING character_set)

Хотя обе функции (**CAST** и **CONVERT**) выполняют одну и ту же задачу, есть несколько различий между ними, о которых нужно знать. Каждая функция лучше подходит для некоторых преобразований, и в некоторых случаях дает лучшую производительность. Функция CONVERT лучше для преобразования значений даты и времени, дробных чисел и денежных значений. С другой стороны, функция CAST более приспособлена к преобразованию десятичных знаков и числовых значений, поскольку функция может сохранять десятичные знаки из исходных выражений. Кроме того, CAST () использует стандарт ANSI и более портируема по сравнению с функцией CONVERT.

Подробное описание: Функции CAST и CONVERT (Transact-SQL)
Для MySQL Функции CAST и CONVERT для MySQL

Примеры:

```
select cast('20-04-2017' as datetime) -- ошибка преобразования
```

```
select cast('20170420' as datetime)
```

MySQL

select convert('2014-02-28 08:14:57', datetime);

T-SQL

```
select 'Студент ' + SURNAME + ' учится на ' + COURSE + ' курсе' from STUDENTS -- ошибка. Попытка конкатенации числа и строки
```

T-SQL

```
select 'Студент ' + SURNAME + ' учится на ' + cast(COURSE as char(1)) + ' курсе' from STUDENTS
```

MySQL

select concat('Студент ', SURNAME, ' учится на ', cast(COURSE as char(1)),' курсе') from STUDENTS

```
select 'Студент ' + SURNAME + ' родился ' + cast(BIRTHDAY as varchar(12)) from STUDENTS
select 'Студент ' + SURNAME + ' родился ' + convert(varchar(12), BIRTHDAY, 105) from STUDENTS

select '0.1415926' + 3 -- ошибка
select convert(float, '0.1415926') + 3
select cast('0.1415926' as float) + 3
```

Особенности функции CONVERT для MS SQL

Функция **CONVERT** в MS SQL Server имеет третий (необязательный) параметр style который:

1. Если аргумент ***expression*** имеет тип ***datetime*** и происходит преобразования в строку, то параметр style определяет шаблон даты для результирующей строки.

Примеры:

```
select convert(varchar, getdate(), 100);
select convert(varchar, getdate(), 101);
select convert(varchar, getdate(), 105);
select convert(varchar, getdate(), 120);
select convert(varchar, getdate(), 127);
```

2. Если происходит конвертирование строки в дату, т.е. первый параметр имеет тип datetime, а второй некоторый строковый тип, то параметр style определяет правильный формат для входной строки.

Примеры:

```
select convert(datetime, '20-01-2019', 101); -- ошибка преобразования
select convert(datetime, '20-01-2019', 105);
```

3. Если параметр ***expression*** имеет тип ***float*** или ***real***, то параметр ***style*** определяет количество знаков после десятичной точки

Примеры:

```
select convert(varchar(25), 0.31415926535879323846E+001, 0);
select convert(varchar(25), 0.31415926535879323846E+001, 1);
select convert(varchar(25), 0.31415926535879323846E+001, 2);
```

Преобразования даты в строку и строки в дату

Преобразование строки в дату (MS SQL)

В зависимости от того, какие установки для даты приняты на сервере, попытка преобразования строки в дату может быть успешной или нет. Например:

```
select cast('20-04-2017' as datetime);
go

SET DATEFORMAT dmy;
select cast('20-04-2017' as datetime);
go
```

SET DATEFORMAT dmy – устанавливает нужный нам формат даты на время сессии. Для того, что бы получить текущие установки формата даты для MS SQL следует выполнить команду:
DBCC useroptions

В случае MS SQL для преобразования строки в дату лучше использовать функцию CONVERT

```
select CONVERT(datetime, '27.12.2018',  104);
select CONVERT(varchar(10), CONVERT(datetime, '27.12.2018',  104),  101);
```

Преобразование даты в строку (MS SQL)

- cast(datetime as varchar)
- convert(datetime, date_string, style)
- format(date_string, format) *(начиная с версии 2012)*

Примеры использования cast и convert

```
select cast(getdate() as varchar);
select convert(varchar, getdate(), 105);
```

Так же, для форматированного вывода даты, для MS SQL можно использовать функцию FORMAT. Функция FORMAT появилась в MS SQL, начиная с версии 2012, и возвращает значение, указанное в формате, языке и региональных параметрах. Для выполнения форматирования значения даты, времени и чисел с учетом локали в виде строк используется функция FORMAT. Для общих преобразований типов данных продолжайте использовать CAST и CONVERT.

Примеры использования функции format в MS SQL:

```
SELECT FORMAT ( getdate(), 'd', 'en-US' ) AS 'US English Result'
      ,FORMAT ( getdate(), 'd', 'en-gb' ) AS 'Great Britain English Result'
      ,FORMAT ( getdate(), 'd', 'de-de' ) AS 'German Result'
      ,FORMAT ( getdate(), 'd', 'zh-cn' ) AS 'Simplified Chinese (PRC) Result';

SELECT FORMAT ( getdate(), 'D', 'en-US' ) AS 'US English Result'
      ,FORMAT ( getdate(), 'D', 'en-gb' ) AS 'Great Britain English Result'
      ,FORMAT ( getdate(), 'D', 'de-de' ) AS 'German Result'
      ,FORMAT ( getdate(), 'D', 'zh-cn' ) AS 'Chinese (Simplified PRC) Result';

SELECT FORMAT( getdate(), 'dd-MMM-yyyy', 'en-US' ) AS 'DateTime Result';
```

Преобразование даты в строку (MySQL)

Выполняется при помощи функций date_format и time_format:

- **date_format**(date, format) - дату в строку;
- **time_format**(time, format) - время в строку;

Ниже приведен список основных элементов форматирования для даты и времени:

- %c - месяц числом;
- %d - день месяца;
- %H - часы (от 0 до 24);
- %h - часы (1 до 12);

- %i - минуты;
- %s - секунды;
- %T - время в формате "hh:mm:ss";
- %Y - год, четыре цифры;
- %y - год, две цифры.

Пример использования функции date_format в MySQL:

```
select   date_format(current_date(), '%d-%m-%Y') as d1
        , date_format(current_date(), '%d-%M-%Y') as d2;
```

Преобразование строки в дату (MySQL)

Для преобразования строки в дату в MySQL используется функция STR_TO_DATE()
Пример:

```
select str_to_date('24-04-2018', '%d-%m-%Y') as dt;
select str_to_date('24-04-2018', '%m-%d-%Y') as dt; -- ошибка
select str_to_date('04-24-2018', '%m-%d-%Y') as dt;
```

Преобразование строки в дату

Для уменьшения проблем преобразования строкового литерала в дату лучше всего использовать формат строкового литерала вида:

'**YYYYMMDD**' ,- где

- YYYY – четыре цифры представляющие год
- MM – две цифры представляющие месяц (01 – январь, 02 – февраль,…, 12 – декабрь)
- DD – две цифры предсталяющие день (01 – первый день,……)

https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/datetime-transact-sql?view=sql-server-2017

Функции для работы со значение NULL

ISNULL (check_expression , replacement_value) - Заменяет значение NULL указанным замещающим значением.

COALESCE (expression [,...n]) - Вычисляет аргументы по порядку и возвращает текущее значение первого выражения, изначально не вычисленного как NULL.

Выражение COALESCE — синтаксический ярлык для выражения CASE. То есть, код COALESCE(*Выражение1,.. .n*) переписывается оптимизатором запросов как следующее выражение CASE:

```
CASE
      WHEN (expression1 IS NOT NULL) THEN expression1
      WHEN (expression2 IS NOT NULL) THEN expression2
...
      ELSE expressionN
END
```

Для **MySQL** следует использовать функцию **IFNULL**

Пример использования:

```
select isnull(birthday) from students;
select ifnull(birthday, cast('19700101' as datetime)) from students;
```

Сравнение COALESCE и ISNULL

Функция ISNULL и выражение COALESCE имеют аналогичные цели, но могут отличаться поведением.

- Поскольку ISNULL — это функция, она вычисляется только один раз. В то время как входные значения для выражения COALESCE могут вычисляться несколько раз.
- Различается определение типа данных результирующего выражения. ISNULL использует тип данных первого параметра, COALESCE следует правилам выражения CASE и возвращает тип данных значения с самым высоким приоритетом.
- Для ISNULL и COALESCE различается допустимость значения NULL для результирующего выражения. Возвращаемое значение ISNULL всегда считается не допускающим значения NULL ***(предполагая, что возвращаемое значение не допускает значения NULL),*** в то время как COALESCE с параметрами, отличными от NULL, считается имеющим значение NULL.

```
select
       convert(varchar(12), birthday, 105) as bd
from STUDENTS

select
       isnull(convert(varchar(12), birthday, 105), 'Неизвестно') as bd
from STUDENTS


select
       COALESCE(birthday,  '19000101') as bd
from STUDENTS
```