

Firebolt documentation Getting started Architecture overview Integrations Account and user management Using the SQL workspace Using the CLI Working with engines Working with tables Using indexes Loading data Working with partitions Working with semi-structured data v Exporting query results Developing with Firebolt General reference SQL commands SQL functions

DECIMAL data type

This topic describes the Firebolt implementation of the DECIMAL data type.

- Overview
 - Default values for precision and scale
 - Precision vs. scale
 - Two options for type coercion between DECIMAL data types
 - Supported functions (Alpha release)

Overview

The DECIMAL data type is an exact numeric data type defined by its precision (total number of digits) and scale (number of digits to the right of the decimal point).

DECIMAL has two optional input parameters: DECIMAL(precision, scale)

The maximum precision is 38. The scale value is bounded by the precision value (scale<=precision).

The precision must be positive, while the scale can be zero or positive.

The NUMERIC data type is a synonym to the DECIMAL data type.

Default values for precision and scale

If the scale is not specified when declaring a column of DECIMAL data type, then it defaults to DECIMAL(precision, 0)

If both the precision and scale are not specified, then it defaults to DECIMAL(38, 0)

Precision vs. scale

If the scale of a value to be stored is greater than the declared scale of the column, the system will round the value to the specified number of fractional digits. If the number of digits to the left of the decimal point exceeds the declared precision, minus the declared scale, an error results.

```
select cast(100.76 as decimal(5,2)); -- 100.76
select cast(100.76 as decimal(5,1)); -- 100.8
select cast(100.76 as decimal(3,1)); -- error
```

Two options for type coercion between DECIMAL data types

P1≠P2 or S1≠S2 (casting required)

Any operation between two decimals with different precision and/or scale requires explicit casting of the result to the desired precision and scale.

```
f(DECIMAL(P1, S1), DECIMAL(P2, S2)) -> CAST(result as DECIMAL(P3, S3))*
```

where P1≠P2 or S1≠S2

P1=P2 and S1=S2 (casting optional)

If the two decimals have the same precision and scale, the result will implicitly cast to the same precision and scale. You can still explicitly cast to any other precision and scale.

```
f(DECIMAL(P1, S1), DECIMAL(P1, S1)) -> DECIMAL(P1, S1))**
```

Exceptions

SUM:

```
SUM(DECIMAL(P1, S1)) = SUM(DECIMAL(38, S1))
```

AVG:

```
AVG(DECIMAL(P1, S1)) = AVG(DECIMAL(38, max(6, S1)))
```

Supported functions (Alpha release)

Operators:

Functions:

SUM, AVG ANY, ANY_VALUE CHECKSUM COUNT

MAX, MAX_BY, MEDIAN, MIN, MIN_BY

CASE, CAST CITY_HASH

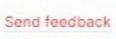
COALESCE, CONCAT

IFNULL, NULLIF, ZEROIFNULL

TRY_CAST

ABS, ROUND

TO_DOUBLE, TO_FLOAT, TO_INT, TO_LONG, TO_STRING



Edit on GitHub

