The 5 Data Types of SQLite

Victor Harris

CIS-150

07/02/2021

The 5 Data Types Of SQLite

There are 5 data types in SQLite: NULL, INTEGER, REAL, TEXT, and BLOB. Unlike other, static data types of other database engines, data types in SQLite are dynamic, and determined by the value itself, rather than its container defined by table creation. Nevertheless, all values stored in an SQLite database will be stored as one of these data types, even it does not match the storing column’s affinity.

**NULL**

Unlike the other data types on this list, NULL is used to define the absence of value. While NULL is often misconceived as being equivalent to a zero value, it is crucial to understand that it is not; NULL cannot have a value of zero, because NULL does not equal anything. It is used to represent the value of fields that are left blank during record creation and to remove values from fields using UPDATE statements. In addition to defining values of fields, the NULL data type can be utilized for data manipulation/analysis using the ‘IS NULL’ AND ‘IS NOT NULL’ operators. When sorting a column that contains values of different data classes, NULL has the lowest value.

**INTEGER**

Simply put, the INTEGER data type is used to store, well, integer values. More specifically, it is used to store whole numbers ranging from 1 to 8 bytes. (-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807). INTEGER values are represented without commas, and do not contain commas, decimal points, or exponents. For compatibility, values in the following syntaxes will be stored with the INTEGER affinity:

* INT
* INTEGER
* TINYINT
* SMALLINT
* MEDIUMINT
* BIGINT
* UNSIGNED BIG INT
* INT2
* INT8

\*It is important to note that while SQLite does accept these syntaxes, it does so to be dynamically compatible with other, more rigidly defined data types of other database engines.

When comparing and sorting INTEGER values to those of other storage classes, they are higher in value than NULL, and compare directly with REAL values numerically.

**REAL**

Values utilizing the REAL data type are stored as 8-byte floating point numbers and include numbers that include a decimal point or exponent. Due to mathematical limitation inherited by the design of floating-point numbers, REAL values are ***approximate,*** though SQLite promised accuracy of the 15 most significant digits of these values. Like INTEGER, the REAL data type is dynamic, and compatible with the following Type Names:

* REAL
* DOUBLE
* DOUBLE PRECISION
* FLOAT

Generally, INTEGER is preferred to REAL when possible because INTEGER values are automatically compressed, as opposed to REAL values which are always stored as 8-byte numbers. Because of this, INTEGER operations are typically faster than similar operations using the REAL data type and (in almost every scenario) will require less storage

**TEXT**

The TEXT data type is used to store values that are alphanumeric strings of text, using the database encoding utf-8, utf-16be, or utf-16le, and are variable in length (virtually unlimited). They are represented literally as a string of characters in single quotes. Although the TEXT data type can store numeric characters, they do not compare directly with values of NUMERIC affinity due to being stored as characters rather than numeric values. However, numeric characters stored as TEXT values *can* be compared using the CAST operator, if compared directly, TEXT values are larger than INTEGER and REAL data types. The text data type is compatible with the following syntaxes:

* CHARACTER(20)
* VARCHAR(255)
* VARYING CHARACTER(255)
* NCHAR(55)
* NATIVE CHARACTER(70)
* TEXT
* CLOB

**BLOB**

The BLOB data type is similar to TEXT in the fact that is it variable in length, but it is also similar to NULL in terms of it’s affinity conversion; neither NULL nor BLOB values are converted, but rather stay as NULL, and BLOB (NONE, no specified datatype, etc.), respectively. Unlike the other data types, BLOB values are copied exactly as provided – hence their name, which stands for Binary Large Object. In terms of comparison and sorting, BLOB is the highest storage class type and will return a higher value than the other data types.

Sources Cited

“Appropriate Uses For SQLite.” *Appropriate Uses For SQLite*, www.sqlite.org/whentouse.html. Accessed 30 June 2021.

“Datatypes In SQLite Version 3.” *SQLite.Org*, www.sqlite.org/datatype3.html. Accessed 30 June 2021.

“Floating Point Numbers.” *Www.Sqlite.Org*, www.sqlite.org/floatingpoint.html. Accessed 30 June 2021.

Kreibich, Jay. “Using SQLite: Small. Fast. Reliable. Choose Any Three.” *Using SQLite*, 1st ed., O’Reilly Media, 2010, pp. 35–39.

Rungta, Krishna. “SQLite Data Types with Example.” *SQLite Data Types*, 28 May 2021, www.guru99.com/sqlite-data-types.html.

---. “SQLite Query: Select, Where, LIMIT, OFFSET, Count, Group By.” *SQLite Query: Select, Where, LIMIT, OFFSET, Count, Group By*, 28 May 2021, www.guru99.com/sqlite-query.html.

“SQLite - Data Type - Tutorialspoint.” *TutorialsPoint*, www.tutorialspoint.com/sqlite/sqlite\_data\_types.htm. Accessed 30 June 2021.

“SQLite Data Types And Its Important Concepts Explained.” *SQLite Tutorial*, 11 Apr. 2020, www.sqlitetutorial.net/sqlite-data-types.

www.techonthenet.com. “SQLite: Data Types.” *TechOnTheNet*, www.techonthenet.com/sqlite/datatypes.php. Accessed 30 June 2021.

Zdila, Martin. “SQLite Dataypes Lengths?” *Stack Overflow*, 13 Oct. 2013, stackoverflow.com/questions/2503797/sqlite-dataypes-lengths.