Michael Watzer 28.05.2017

# gen/sqlvalidation.php

## MySQL

### Supported data types

| Name | Storage requirements(in byte) |
|---|---|
| INTEGER | 4 |
| INT | 4 |
| TINYINT | 1 |
| SMALLINT | 2 |
| MEDIUMINT | 3 |
| BIGINT | 8 |
| FLOAT | 4 |
| DOUBLE | 8 |
| DECIMAL | ~ 29 |
| DATE | 3 |
| DATETIME | 11 |
| TIME | 6 |
| YEAR | 1 |
| CHAR | VARIABLE |
| VARCHAR | VARIABLE |
| TEXT | VARIABLE |
| TINYTEXT | VARIABLE |
| MEDIUMTEXT | VARIABLE |
| LONGTEXT | VARIABLE |

### Build up of given array

```
M = {name, datatype, size, index, null, autoincrement, selected, default}
name          = Name of column
datatype      = Data type of column
size          = Size of variable length data type
index         = 1(true), -1(false)
autoincrement = 1(true), -1(false)
selected      = Considered in the SQL-Code generation
default       = Default value
```

Michael Watzer 28.05.2017

We define
$Mi_1$ as the data type of the column and
$Mi_2$ as the size of the variable length data type and
$Mi_3$ as the null field, whereas one equals true

## Row length calculation

The following mathematical calculation was tested on the MySQL version 5.5.x. Currently the engine and encoding type are InnoDB and utf8, so every character represented by utf8 is three bytes long.

### Sets

A = Entire set of columns used in the CREATE TABLE statement.
Av = {CHAR, VARCHAR, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT}
Set of variable data types.

Avl = {CHAR, VARCHAR}
Set of variable data types where a length is given.

So the following rule is true

- Avl c Av

### Variable row length calculation

$$S(M) = \sum_{i=0;x=0}^{n} \begin{cases} \text{if } Mi_1 \in Avl \mid x := x + (Mi_2 * 3)) \\ \text{if } Mi_1 \notin Av \mid x := x + Mi_2)) \end{cases}$$ whereas x is returned

### Null entry calculation

$$N(M) = \sum_{i=0;x=7}^{n} (Mi \in A \bigwedge Mi_3 = 1 \mid x := x + 1)$$, whereas x is returned

### Variable data type calculation

$$V(M) = \sum_{i=0;x=0}^{n} (Mi_1 \in Av \mid x := x + 1)$$, whereas x is returned

### Calculation

The row length calculation can be declared as

$$Rl(A) := 1 + (S(A)) + ((N(A)) / 8) + (V(A))$$

Michael Watzer 28.05.2017

## Data types and their default value

| Name | Value |
|---|---|
| INT/INTEGER | >= -2147483648<br><= 2147483647 |
| TINYINT | >= -128<br><= 127 |
| SMALLINT | >= -32768<br><= 32767 |
| MEDIUMINT | >= -8388608<br><= 8388607 |
| BIGINT | >= -9223372036854775808<br><= 9223372036854775807 |
| YEAR | If one or two decimal places:<br>>= 0<br><= 99<br>If four decimal places:<br>>= 1901<br>>= 2155 |
| VARCHAR | If size given:<br>String length <= size<br>If no size given:<br>String length <= 21844 |
| CHAR | If size given:<br>String length <= size<br>If no size given:<br>String length <= 2^8 - 1 |
| TEXT | String length <= 2^16 - 2 |
| TINYTEXT | String length <= 2^8 - 1 |
| MEDIUMTEXT | String length <= 2^24 - 3 |
| LONGTEXT | String length <= 2^32 - 4 |
| DATE | YYYY-MM-DD<br>Whereas the range of YYYY is between 1000 and 9999.<br>MM can be represented with a leading zero, e.g 01.<br>Allowed values are 01 up to 12.<br>The range of DD depends on the given year/month.<br>To ensure a valid input the build-in PHP function checkdate is used. See:<br>*http://php.net/manual/de/function.checkdate.php* |
| DATETIME | YYYY-MM-DD [HH24:MM:SS]<br>The date part is validated with the DATE-Type rule.<br>The TIME part can be omitted.<br>HH24: >= 0, <= 23<br>MM and SS: >= 0, <= 59 |

| | |
|---|---|
| TIME | HH24:MM:SS<br>The single TIME-Type can have a larger range than the TIME part in the DATETIME-Type.<br>The reason according to MySQL is that the TIME-Type can be used to measure TIME between two events.<br>HH24: >= -838, <= 838<br>MM and SS: >= 0, <= 59 |
| FLOAT<br>DOUBLE<br>DECIMAL | m.d<br>FLOAT, DOUBLE MAXSIZE<br>m → 255<br>d → 30<br>DECIMAL MAXSIZE<br>m → 65<br>d → 30<br>If size given:<br>SIZE = SIZE<br>If no size given:<br>SIZE = MAXSIZE<br>m is the total length of the number, whereas d is the number of allowed decimal places.<br>So we validate a floating type with:<br>number length <= m<br>number of decimal places <= d<br>E.g<br>SIZE = 6, 4<br>18.0102 → VALID<br>1.01010 → INVALID |

# SQLite

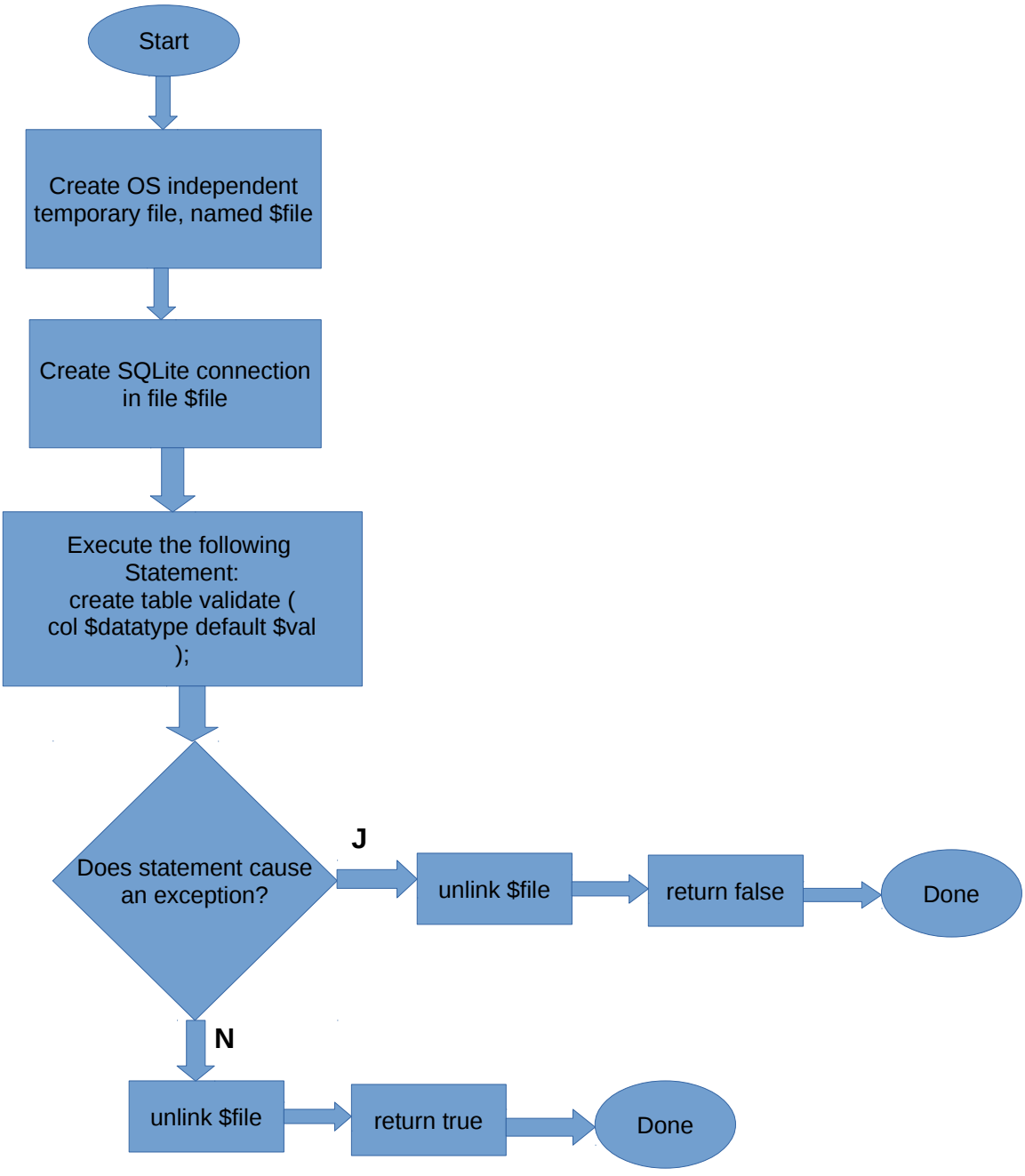
## Supported data types

- INTEGER
- INT
- TINYINT
- SMALLINT
- MEDIUMINT
- BIGINT
- FLOAT
- DOUBLE
- REAL
- DATE

- DATETIME
- DECIMAL
- CHAR
- VARCHAR
- TEXT

## Data types and their default value

To ensure a valid default value a native validation is used.
To be exact the following program unit is used:



Michael Watzer 28.05.2017

Reason why we use a native validation:

*„Most SQL database engines (every SQL database engine other than SQLite, as far as we know) uses static, rigid typing. With static typing, the datatype of a value is determined by its container - the particular column in which the value is stored.*

*SQLite uses a more general dynamic type system. In SQLite, the datatype of a value is associated with the value itself, not with its container. The dynamic type system of SQLite is backwards compatible with the more common static type systems of other database engines in the sense that SQL statements that work on statically typed databases should work the same way in SQLite. However, the dynamic typing in SQLite allows it to do things which are not possible in traditional rigidly typed databases.“*

– *https://sqlite.org/datatype3.html*