02 常用数据类型与使用建议

更新时间：2020-03-09 11:04:57

> 虚心使人进步，骄傲使人落后。——毛泽东

在日常使用 MySQL 的过程中，一定会根据数据类型的限制、特性有所取舍。MySQL 中可选的数据类型有很多，每一种数据类型都会有其使用限制与适合的使用场景，想把它们都理清楚、说明白并不是一件简单的事。但是，这些知识点又会极大的影响我们日常的工作使用。所以，有必要对常用的数据类型做思考总结。

1 关于数据类型的说明

数据类型定义了 MySQL 列中可以存储什么数据以及当前数据存储的基本规则。从 MySQL 系统的角度出发，则是为了方便对数据分类，能够使用统一的方式进行管理，更有效的利用有限空间的一种手段。下面，我会介绍 MySQL 数据类型的分类以及 MySQL 自身提供的对数据类型的帮助说明。

1.1 数据类型分类

通常，我们会将 MySQL 的数据类型分为四类，即字符串、日期 / 时间、数值以及二进制。显然，根据这些分类的名称可以知道，分类是按照存储数据的类型来做的。那么，这些分类中又包含了哪些数据类型呢 ？

- 字符串类型：以 char、varchar、text 为代表，用于存储字符、字符串数据
- 日期 / 时间类型：以 date、time、datetime、timestamp 为代表，用于存储日期或时间，这种数据类型也是比较难抉择的
- 数值类型：以 tinyint、int、bigint、float、double、decimal 为代表，用于存储整数或小数
- 二进制类型：以 tityblob、blob、mediumblob、longblob 为代表，用于存储二进制数据，适用场景最为受限

最后，需要说明，对数据类型的分类并不是绝对的，这取决于对存储数据的限制程度。例如对于数值类型又可以再去细分为整数型（int、bigint 等）、浮点型（float、double 等）、定点型（decimal 等）。所以，并不需要把过多的精力花在类型分类上，更多的是应该搞清楚这些类型怎么用，又为什么这样用。

1.2 MySQL 的 help 命令

对于平时写代码的你来说，Linux/Unix 环境一定不会陌生，当然，也就对 man 和 help 这样的命令不会陌生了。类似于这样的 “帮助命令” 在 MySQL 中也是有的。例如，你想知道 int 这种数据类型的使用范围，可以执行命令：

```
mysql> help int
Name: 'INT'
Description:
INT[(M)] [UNSIGNED] [ZEROFILL]

A normal-size integer. The signed range is -2147483648 to 2147483647.
The unsigned range is 0 to 4294967295.

URL: https://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html
```

可以看到，help 打印了 int 数据类型的描述信息以及官方文档的链接地址。这对于学习使用数据类型来说，是非常方便的。当然，我们也可以在 help 后面加上 char、varchar 等等 MySQL 支持的数据类型。

此时，你可能会有疑问：这些打印的信息是从哪里来的 ？难道也是保存在 MySQL 表中的吗 ？确实，正如猜测的那样，MySQL 提供了 4 张表用于保存帮助信息（help 语法打印的即为帮助信息）。这些表位于 mysql 系统字典库中，且表名都以 help_ 开头。如下所示：

```
mysql> show tables from mysql where Tables_in_mysql like 'help_%';
+-----------------+
| Tables_in_mysql |
+-----------------+
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
+-----------------+
```

这些表是在数据库初始化时通过内建脚本创建而成，其中：

- help_category：存储关于帮助主题类别的信息
- help_keyword：存储与帮助主题相关的关键字信息
- help_relation：存储帮助关键字信息和主题信息之间的映射
- help_topic：存储帮助主题的详细内容

由此，可以知道，我们之前的 help int 信息来自于 mysql.help_topic 表中，也就可以通过查询表记录信息来获取帮助信息了。如下所示：

```
-- \G 指示 MySQL 以列格式打印结果信息
mysql> SELECT * FROM mysql.help_topic WHERE name = 'int'\G
*************************** 1. row ***************************
  help_topic_id: 13
        name: INT
help_category_id: 2
    description: INT[(M)] [UNSIGNED] [ZEROFILL]

A normal-size integer. The signed range is -2147483648 to 2147483647.
The unsigned range is 0 to 4294967295.

URL: https://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html


      example:
        url: https://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html
```

关于其他的“帮助表”这里不再过多介绍，有兴趣的同学可以自行查询 MySQL 官网或其他渠道了解信息。

2 常用数据类型解读及使用建议

如果你在工作中细心观察，你会发现，其实常用的数据类型并不会很多。下面，我将会按照之前的分类对日常工作中最常用的数据类型进行解读，同时也会说明它们各自的适用场景与使用时的建议。

2.1 字符串类型

2.1.1 char

char 数据类型用于定义一个固定长度的字符串，长度范围处于 1 ~ 255 之间，且必须是在创建表时指定。它有一个特殊的情况是，存储字符串时，如果未达到指定长度，则会使用空格填充到指定长度。所以，如果我们想要存储不同记录的字符串长度差别较大，会造成较大的空间浪费。

根据对 char 类型的描述可以知道，当我们需要存储一些长度固定的数据列时，使用 char 是非常合适的。例如：手机号码、身份证号等等。

2.1.2 varchar

相对于 char 来说，varchar 的“出场率”要稍微高一些。它定义了一个可变长度的字符串，创建时指定它所允许的最大长度。例如，如果创建时声明了 varchar (x)，则只能存储不超过 x 个字符的数据，且 x 的最大值是 65535。

对于长度不固定的数据列，使用 varchar 就是最合适的。例如：姓名、邮箱地址等等。

char 和 varchar 是非常相似且常见的字符串类型，想要把它们用对、用好，不仅要能够理解它们各自的含义、特性，还要知道它们在使用上的区别：

- 定义了 char (x)，如果存入的字符个数小于 x，则以空格填充，查询时再将空格去掉（类似于 trim 操作）。所以，char 类型存储的字符串末尾不能有空格，而 varchar 则没有这一限制
- char (x) 长度是固定的，不论存入什么，都会占用 x 个字节。但是 varchar 占用的字节数是存入的字符数 + 1（x <= 255）或 + 2（x > 255）
- char 由于长度固定，不需要考虑边界问题，检索速度要快于 varchar

2.1.3 tinytext、text、mediumtext、longtext

这是 MySQL 提供的四类文本数据类型，它们都属于变长字符串，最大的区别是存储空间的不同，其中：

- tinytext：最大长度是（2^8 - 1）个字符
- text：最大长度是（2^16 - 1）个字符
- mediumtext：最大长度是（2^24 - 1）个字符
- longtext：最大长度是（2^32 - 1）个字符

最简单的对文件数据类型的理解是：当我们要存储的数据量比较大，就应该考虑使用文本。这里，我建议当你的数据量超过 500 个字符时，就应该考虑使用文本。另外，文本类型不能有默认值，且在创建索引时需要指定前多少个字符。

2.2 日期 / 时间类型

2.2.1 date

正如这种数据类型的名称一样，它用于存储日期，存储范围是 '1000-01-01' 到 '9999-12-31'。这种数据类型比较简单，但同时适用场景也比较有限，因为它只能存储 "年月日"。比较常见的用途是存储出生日期。

2.2.2 time

它用于存储时间，不仅可以表示一天中的时间，也可以用于表示两个时间的时间间隔。它的取值范围是 '-838:59:59' to '838:59:59'。乍看起来，它的小时取值太特殊了，正常不应该是 [0, 23] 吗 ？这是因为 time 可以表示特殊的时间间隔，MySQL 将 time 的小时范围扩大了，而且支持负值。

除了基本的存储一天中的时间之外，time 允许以 "D HH:MM:SS" 的格式存储。其中，D 的取值是 0 ~ 34。如果要存储时间间隔，time 则会以（时间间隔 * 小时）作为小时进行存储。它的计算公式是：D * 24 + HH。例如，插入了 "2 19:20:00"，相当于插入 "67:20:00"。

2.2.3 datetime

日期与时间的组合格式，取值范围是 '1000-01-01 00:00:00.000000' 到 '9999-12-31 23:59:59.999999'。它是最常见，用途最广的数据类型。例如：存储数据插入时间、订单完成时间等等。

2.2.4 timestamp

同样用于存储日期时间数据，与 datetime 存储的数据格式是一样的，它的取值范围是：'1970-01-01 00:00:01.000000' UTC 到 '2038-01-19 03:14:07.999999' UTC。它与 datetime 的主要区别在于时间范围要小一些。

另外，timestamp 是与时区相关的，能够反映 "当前时间"。当插入时间时，会先转换为本地时区后再存储；查询时间时，会转换为本地时区后再显示。所以，不同时区的人看到的同一时间是不一样的。

在 MySQL 表中存储时间（可以是日期、时间或日期时间）是非常常见的需求，但是如何合理的选择数据类型却也是个难题。这里我给出一个建议：通常 datetime 是最佳选择。理由如下：

- 时间范围跨度足够大，能够满足所有的时间需求
- 即使是只用于存储日期或时间，也可以存储日期时间，只需要在代码中处理即可。避免将来需求变更时对数据表的 Schema 有所变动。

2.3 数值类型

2.3.1 整数类型

MySQL 主要支持 5 个整数类型：tinyint、smallint、mediumint、int、bigint。这些数据类型我们基本上认为它们有共同的特性，不同之处只在于存储空间，即存储数值的取值范围。同时，在定义时可以使用 UNSIGNED 关键字规定字段只保存正值。下面，我将这几种整数类型的特性用表格展示出来。

数据类型	占据空间	范围（有符号）	范围（无符号）	描述
tinyint	1 个字节	-2^7 - 2^7 - 1	0 - 255	小整数值
smallint	2 个字节	-2^15 - 2^15 - 1	0 - 65535	大整数值
mediumint	3 个字节	-2^23 - 2^23 - 1	0 - 16777215	大整数值
int	4 个字节	-2^31 - 2^31 - 1	0 - 4294967295	大整数值
bigint	8 个字节	-2^63 - 2^63-1	0 - 18446744073709551615	极大整数值

由于这几种数据类型除了取值范围不同之外，并没有其他的不同，所以，在使用上，根据需要选择 “足够大” 的空间就可以了。另外，关于整数类型还有一个特性：显示宽度。例如，我们在定义 Schema 时，常常会看到类似这样的写法：

```
`a` bigint(20) NOT NULL COMMENT 'a',
`b` int(11) NOT NULL COMMENT 'b'
```

其中，20 和 11 就是可选的显示宽度，这会让 MySQL 对 SQL 标准进行扩展，当从数据库检索一个值时，可以把这个值延长到指定的宽度。例如，这里的 b 定义的类型为 int (11)，就可以保证 b 这一列少于 11 个字符宽度时自动使用空格填充。但同时，需要注意，定义宽度并不会影响字段的大小和存储值的取值范围。

2.3.2 浮点类型

MySQL 支持两个浮点类型：float、double。其中，float 用于表示单精度浮点数值，占用 4 个字节；double 用于表示双精度浮点数值，占用 8 个字节。因为它们只能保存近似值（不精确的值），所以，通常也叫做非标准类型。float 相较于 double 类型来说，由于占据的空间小，精度较低，取值范围也相对较小。它们的定义格式及说明如下：

- float (M, D)：其中 M 定义显示长度，D 定义小数位数。但是它们是可选的，且默认值是 float (10, 2)，2 是小数的位数，10 是数字的总长（包括小数）。它的小数精度可以到 24 个浮点。
- double (M, D)：M 和 D 的含义与 float 是相同的，默认值是 double (16, 4)。它的小数精度可以达到 53 位。

2.3.3 定点类型

MySQL 中的 decimal 被称为定点数据类型，由于它保存的是精确值，所以它通常用于精度要求非常高的计算中。另外，也可以利用 decimal 去保存比 bigint 还要大的整数值。

需要知道，CPU 并不支持对 decimal 的直接计算，而是 MySQL 自身实现了对 decimal 的高精度计算。底层存储方面，MySQL 将 decimal 类型的数字使用二进制字符串存储，每 4 个字节可以存储 9 个数字。假如我们定义了 decimal (18, 9)：

- 则代表不包含小数点的数字总数（整数位数 + 小数位数）位数是 18，不指定的情况下默认是 10
- 9 则代表小数的位数，如果不指定，默认是 0

由于小数点两边各有 9 个数字，所以占据 2 * 4 = 8 个字节，小数点自身占用一个字节，最终，decimal (18, 9) 一共占用 9 个字节。需要注意，如果存储的位数不够，则小数末尾会补零。但是，如果超出了声明的位数，则会报错。

由于 decimal 需要比较大的空间和计算开销，它的计算效率也就没有 float 和 double 那么高，所以应该只有要求精确计算的场景下才考虑去使用 decimal。

2.4 二进制类型

二进制数据类型理论上可以存储任何数据，可以是文本数据，也可以存储图像或者其他多媒体数据。二进制数据类型相对于其他的数据类型来说，使用频率是比较低的。MySQL 一共提供了四种二进制类型：tityblob、blob、mediumblob、longblob，它们的区别只在于存储范围的不同。

- tityblob：最大支持 255 字节
- blob：最大支持 64KB
- mediumblob：最大支持 16MB
- longblob：最大支持 4GB

需要注意，虽然 MySQL 提供并支持大文件存储，但是这样会急剧降低数据库的性能。所以，应该谨慎使用这些数据类型，能不用的情况下尽量不用。

3 数据类型选择与使用上的技巧与建议

在我们日常使用 MySQL 的过程中，或多或少都会积累一些经验。下面，我将总结我在工作中对数据类型选择的一些技巧与建议。但是，需要知道，这些建议并不一定适用于所有的情况。在做实际的选择时，我们不仅要考虑这些技巧，也要对应到具体的需求。

3.1 使用 NOT NULL，且带有 COMMENT

这个建议适用于所有的数据类型，MySQL 在索引值为 NULL 的列时，需要额外的存储空间，所以，相对于 NOT NULL 来说，NULL 会占用更多的空间。另外，在进行比较和计算时，MySQL 要对 NULL 值做特别的处理，使用效率较低。

COMMENT 用于定义列的注释信息，就好像我们在写代码一样，把重要的或者不易理解的地方，加上一些注释，方便以后查阅。

3.2 使用存储需要的最小数据类型

这里所说的最小数据类型并不是直接选择最小的，而是在满足需求的同时选择最小的。例如，要存储事件状态，可以选择 tinyint；要存储班级人数，可以选择 smallint 等等。关于最小数据类型，它有两大优势：

- 越小的数据类型占用的磁盘、内存、CPU 缓存都会更小，存取速度也会更快
- 小的数据类型建立索引时所需要的空间也相对较小，这样一页中所能存储的索引节点数量也就越多，遍历时 IO 次数就会越少，索引的性能也就越好

3.3 选择简单的数据类型

这里的“简单”二字听上去会比较奇怪，我以一个例子去说明。假如说我想在一列中存储 10、100、201 这样的数据，我们可以选择使用 int 或 varchar 来存储。但是整型要比字符型的操作复杂度小太多，那么，选择整型（例如 int）就是最简单的数据类型。

3.4 存储小数直接选择 decimal

虽然我并不建议在数据库中存储小数，但是，在一些场景中小数不可避免，最常见的例子就是订单的金额。由于小数本身在计算时就很复杂，而且很多时候你需要去考虑精度问题。所以，最直接的方式就是把这种管理交给数据库。

这里我提出一个扩展建议，也就是不要在数据库中存储小数。那么，假如订单的精度到分（元、角、分）级别，我们可以考虑在存储时，把数据值 * 100 再去存储。之后，在代码中处理分的逻辑，也就是自己去控制处理小数的精度问题。

3.5 尽量避免使用 text 和 blob

MySQL 内存临时表并不支持 text、blob 这样的大数据类型，如果查询时包含有这样的数据，则排序操作必须使用磁盘临时表，性能会下降很多。而且对于这种数据，MySQL 还要做二次查询（因为 MySQL 实际保存的是指针，而不是真实数据），会使 SQL 性能变得很差。

但是，也并不是说我们一定就不能用 text 和 blob。如果确实有需求需要使用这样的数据类型，那么在查询时一定不要直接 SELECT *，而是取出需要的列。这样 MySQL 就不会去主动查询这些数据列，也是提高性能的一种惯用手段。

最后，还需要注意，因为 MySQL 对索引长度的限制，text 类型只能用到前缀索引，并且由于存储的是指针，text 列上不能有默认值。

4 总结

数据类型是 MySQL 的基础，看起来也比较简单，但常常也就是觉得简单才会忽略它们的特性与限制。可以肯定的说，想要选择正确的、合理的数据类型并不是一件简单的事。不过，也并不需要追求完美的选型。能够解决实际的问题，或多或少存在一些瑕疵，当然也是可以接受的。先去学习并理解，再去大胆的使用，遇到瓶颈了再回过头仔细分析问题，并解决掉，这就是很好的学习方法。

5 问题

将时间转换为时间戳，并使用 int 或者 bigint 类型去存储，你觉得这样可行吗 ？

大多数时候，我们会选择将主键设置为 bigint 数据类型，你知道这是为什么吗 ？

6 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档

}

01 开篇词-你为什么要学习 MySQL？

03 Schema 设计规范是什么样的？