常见的 MySQL 数据类型

数据类型（data_type）是指系统中所允许的数据的类型。MySQL 数据类型定义了列中可以存储什么数据以及该数据怎样存储的规则。

数据库中的每个列都应该有适当的数据类型，用于限制或允许该列中存储的数据。例如，列中存储的为数字，则相应的数据类型应该为数值类型。

如果使用错误的数据类型可能会严重影响应用程序的功能和性能，所以在设计表时，应该特别重视数据列所用的数据类型。更改包含数据的列不是一件小事，这样做可能会导致数据丢失。因此，在创建表时必须为每个列设置正确的数据类型和长度。

MySQL 的数据类型一般分为以下 5 种，分别是整数类型、浮点数类型和定点数类型、日期和时间类型、字符串类型、二进制类型等，整数类型和浮点数类型可以统称为数值数据类型。

1）数值类型

整数类型包括 TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT。

2）浮点数类型和定点数类型

浮点数类型包括 FLOAT 和 DOUBLE，定点数类型为 DECIMAL。

3）日期/时间类型

包括 YEAR、TIME、DATE、DATETIME 和 TIMESTAMP。

4）字符串类型

包括 CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM 和 SET 等。

5）二进制类型

包括 BIT、BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。

定义字段的数据类型对数据库的优化是十分重要的。

4.1.1 整数类型

整数类型又称数值型数据，数值型数据类型主要用来存储数字。

MySQL 提供了多种数值型数据类型，不同的数据类型提供不同的取值范围，可以存储的值范围越大，所需的存储空间也会越大。

MySQL 主要提供的整数类型有 TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT，其属性字段可以添加 AUTO_INCREMENT 自增约束条件。下表中列出了 MySQL 中的数值类型。

表 4-1 整数类型

类型名称	说明	存储需求
TINYINT	很小的整数	1 个字节
SMALLINT	小的整数	2 个字节
MEDIUMINT	中等大小的整数	3 个字节
INT (INTEGHR)	普通大小的整数	4 个字节
BIGINT	大整数	8 个字节

从上表中可以看到，不同类型的整数存储所需的字节数不相同，占用字节数最小的是 TINYINT 类型，占用字节最大的是 BIGINT 类型，占用的字节越多的类型所能表示的数值范围越大。以 int 为例，占用的存储字节数是 4 个，即 4*8=32 位，2 的 32 次方，无符号的最大能达到 4 亿多。

表 4-2 整数类型表示范围

类型名称	说明	存储需求
TINYINT	-128～127	0 ～255
SMALLINT	-32768～32767	0～65535
MEDIUMINT	-8388608～8388607	0～16777215
INT (INTEGER)	-2147483648～2147483647	0～4294967295
BIGINT	-9223372036854775808～9223372036854775807	0～18446744073709551615

注意：后面的是默认显示宽度。

其他整型数据类型也可以在定义表结构时指定所需的显示宽度，如果不指定，则系统为每一种类型指定默认的宽度值。不同的整数类型有不同的取值范围，并且需要不同的存储空间，因此应根据实际需要选择最合适的类型，这样有利于提高查询的效率和节省存储空间。

4.1.2 浮点和定点数类型

MySQL 中使用浮点数和定点数来表示小数。

浮点类型有两种，分别是单精度浮点数（FLOAT）和双精度浮点数（DOUBLE）；定点类型只有一种，就是 DECIMAL。

浮点类型和定点类型都可以用(M,D)来表示，其中 M 称为精度，表示总共的位数；D 称为标度，表示小数的位数。

浮点数类型的取值范围为 M（1～255）和 D（1～30，且不能大于 M-2），分别表示显示宽度和小数位数。M 和 D 在 FLOAT 和 DOUBLE 中是可选的，FLOAT 和 DOUBLE 类型将被保存为硬件所支持的最大精度。DECIMAL 的默认 D 值为 0、M 值为 10。

表 4-3 浮点和定点数类型

类型名称	说明	存储需求
FLOAT	单精度浮点数	4 个字节
DOUBLE	双精度浮点数	8 个字节
DECIMAL (M, D)，DEC	压缩的“严格”定点数	M+2 个字节

DECIMAL 类型不同于 FLOAT 和 DOUBLE。DOUBLE 实际上是以字符串的形式存放的，DECIMAL 可能的最大取值范围与 DOUBLE 相同，但是有效的取值范围由 M 和 D 决定。如果改变 M 而固定 D，则取值范围将随 M 的变大而变大。

从上表中可以看到，DECIMAL 的存储空间并不是固定的，而由精度值 M 决定，占用 M+2 个字节。

FLOAT 类型的取值范围如下：

有符号的取值范围：-3.402823466E+38～-1.175494351E-38。

无符号的取值范围：0 和-1.175494351E-38～-3.402823466E+38。

DOUBLE 类型的取值范围如下：

有符号的取值范围：-1.7976931348623157E+308～-2.2250738585072014E-308。

无符号的取值范围：0 和-2.2250738585072014E-308～-1.7976931348623157E+308。

提示：不论是定点还是浮点类型，如果用户指定的精度超出精度范围，则会四舍五入进行处理。

FLOAT 和 DOUBLE 在不指定精度时，默认会按照实际的精度（由计算机硬件和操作系统决定），DECIMAL 如果不指定精度，默认为（10，0）。

浮点数相对于定点数的优点是在长度一定的情况下，浮点数能够表示更大的范围；缺点

是会引起精度问题。

最后再强调一下：在 MySQL 中，定点数以字符串形式存储，在对精度要求比较高的时候（如货币、科学数据），使用 DECIMAL 的类型比较好，另外两个浮点数进行减法和比较运算时也容易出问题，所以在使用浮点数时需要注意，并尽量避免做浮点数比较。

4.1.3 日期时间类型

MySQL 中表示日期的数据类型有：YEAR、TIME、DATE、DTAETIME、TIMESTAMP。当只记录年信息的时候，可以只使用 YEAR 类型。

每一个类型都有合法的取值范围，当指定确定不合法的值时，系统将“零”值插入数据库中。

下表中列出了 MySQL 中的日期与时间类型。

表 4-4 日期时间类型

类型名称	日期格式	日期范围	存储需求
YEAR	YYYY	1901 ~ 2155	1 个字节
TIME	HH:MM:SS	-838:59:59 ~ 838:59:59	3 个字节
DATE	YYYY-MM-DD	1000-01-01 ~ 9999-12-3	3 个字节
DATETIME	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59	8 个字节
TIMESTAMP	YYYY-MM-DD HH:MM:SS	1980-01-01 00:00:01 UTC ~ 2040-01-19 03:14:07 UTC	4 个字节

- YEAR 类型

YEAR 类型是一个单字节类型，用于表示年，在存储时只需要 1 个字节。可以使用各种格式指定 YEAR，如下所示：

以 4 位字符串或者 4 位数字格式表示的 YEAR，范围为’1901’～’2155’。输入格式为’YYYY’或者 YYYY，例如，输入’2010’或 2010，插入数据库的值均为 2010。

以 2 位字符串格式表示的 YEAR，范围为’00’到’99’。’00’～’69’和’70’～’99’范围的值分别被转换为 2000～2069 和 1970～1999 范围的 YEAR 值。’0’与’00’的作用相同。插入超过取值范围的值将被转换为 2000。

以 2 位数字表示的 YEAR,范围为 1～99。1～99 和 70～99 范围的值分别被转换为 2001～2069 和 1970～1999 范围的 YEAR 值。注意，在这里 0 值将被转换为 0000，而不是 2000。

提示：两位整数范围与两位字符串范围稍有不同。例如，插入 3000 年，程序员可能会使用数字格式的 0 表示 YEAR，实际上，插入数据库的值为 0000，而不是所希望的 3000。只有使用字符串格式的’0’或’00’，才可以被正确解释为 3000，非法 YEAR 值将被转换为 0000。

- TIME 类型

TIME 类型用于只需要时间信息的值，在存储时需要 3 个字节。格式为 HH:MM:SS。HH 表示小时，MM 表示分钟，SS 表示秒。

TIME 类型的取值范围为-838:59:59～838:59:59，小时部分如此大的原因是 TIME 类型不仅可以用于表示一天的时间（必须小于 24 小时），还可能是某个事件过去的时间或两个事件之间的时间间隔（可大于 24 小时，或者甚至为负）。

可以使用各种格式指定 TIME 值，如下所示。

’D HH:MM:SS’格式的字符串。还可以使用这些“非严格”的语法：’HH:MM:SS’、’HH:MM’、’D HH’或’SS’。这里的 D 表示日，可以取 0～34 之间的值。在插入数据库时，D 被转换为小

时保存，格式为“D*24+HH”。

’HHMMSS’格式、没有间隔符的字符串或者 HHMMSS 格式的数值，假定是有意义的时间。例如，’101112’ 被理解为’10:11:12’，但是’106112’是不合法的（它有一个没有意义的分钟部分），在存储时将变为 00:00:00。

提示：为 TIME 列分配简写值时应注意：如果没有冒号，MySQL 解释值时，假定最右边的两位表示秒。（MySQL 解释 TIME 值为过去的时间而不是当前的时间）。例如，程序员可能认为’1112’和 1112 表示 11:12:00（即 11 点 12 分钟），但 MySQL 将它们解释为 00:11:12（即 11 分 12 秒）。同样’12’和 12 被解释为 00:00:12。相反，TIME 值中如果使用冒号则肯定被看作当天的时间，也就是说，’11:12’表示 11:12:00，而不是 00:11:12。

- DATE 类型

DATE 类型用于仅需要日期值时，没有时间部分，在存储时需要 3 个字节。日期格式为’YYYY-MM-DD’，其中 YYYY 表示年，MM 表示月，DD 表示日。

在给 DATE 类型的字段赋值时，可以使用字符串类型或者数字类型的数据插入，只要符合 DATE 的日期格式即可。如下所示：

以’YYYY-MM-DD’或者’YYYYMMDD’字符中格式表示的日期，取值范围为’1000-01-01’～’9999-12-3’。例如，输入’2015-12-31’或者’20151231’，插入数据库的日期为 2015-12-31。

以’YY-MM-DD’或者’YYMMDD’字符串格式表示日期，在这里 YY 表示两位的年值。MySQL 解释两位年值的规则：’00～69’范围的年值转换为’2000~2069’，’70~99’范围的年值转换为’1970～1999’。例如，输入’15-12-31’，插入数据库的日期为 2015-12-31；输入’991231’，插入数据库的日期为 1999-12-31。

以 YYMMDD 数字格式表示的日期，与前面相似，00~69 范围的年值转换为 2000～2069，80～99 范围的年值转换为 1980～1999。例如，输入 151231，插入数据库的日期为 2015-12-31，输入 991231，插入数据库的日期为 1999-12-31。

使用 CURRENT_DATE 或者 NOW()，插入当前系统日期。

提示：MySQL 允许“不严格”语法：任何标点符号都可以用作日期部分之间的间隔符。例如，’98-11-31’、’98.11.31’、’98/11/31’和’98@11@31’ 是等价的，这些值也可以正确地插入数据库。

- DATETIME 类型

DATETIME 类型用于需要同时包含日期和时间信息的值，在存储时需要 8 个字节。日期格式为’YYYY-MM-DD HH：MM：SS’，其中 YYYY 表示年，MM 表示月，DD 表示日，HH 表示小时，MM 表示分钟，SS 表示秒。

在给 DATETIME 类型的字段赋值时，可以使用字符串类型或者数字类型的数据插入，只要符合 DATETIME 的日期格式即可，如下所示。

以’YYYY-MM-DD HH：MM：SS’或者’YYYYMMDDHHMMSS’字符串格式表示的日期，取值范围为’1000-01-01 00:00:00’～’9999-12-3 23:59:59’。例如，输入’2014-12-31 05:05:05’或者’20141231050505’，插入数据库的 DATETIME 值都为 2014-12-31 05:05:05。

以’YY-MM-DD HH:MM:SS’或者’YYMMDDHHMMSS’字符串格式表示的日期，在这里 YY 表示两位的年值。与前面相同，’00～79’范围的年值转换为’2000～2079’，’80～99’范围的年值转换为’1980～1999’。例如，输入’14-12-31 05:05:05’，插入数据库的 DATETIME 为 2014-12-31 05:05:05；输入 141231050505，插入数据库的 DATETIME 为 2014-12-31 05:05:05。

以 YYYYMMDDHHMMSS 或者 YYMMDDHHMMSS 数字格式表示的日期和时间。例如，输入 20141231050505，插入数据库的 DATETIME 为 2014-12-31 05:05:05；输入 140505050505，插入数据库的 DATETIME 为 2014-12-31 05:05:05。

提示：MySQL 允许“不严格”语法，任何标点符号都可用作日期部分或时间部分之间的

间隔符。例如，'98-12-31 11:30:45'、'98.12.31 11+30+35'、'98/12/31 11*30*45' 和 '98@12@31 11^30^45' 是等价的，这些值都可以正确地插入数据库。

- TIMESTAMP 类型

TIMESTAMP 的显示格式与 DATETIME 相同，显示宽度固定在 19 个字符，日期格式为 YYYY-MM-DD HH:MM:SS,在存储时需要 4 个字节。但是 TIMESTAMP 列的取值范围小于 DATETIME 的取值范围，为'1970-01-01 00:00:01'UTC～'2038-01-19 03:14:07'UTC。在插入数据时，要保证在合法的取值范围内。

提示：协调世界时（英：Coordinated Universal Time，法：Temps Universel Coordonné）又称为世界统一时间、世界标准时间、国际协调时间。英文（CUT）和法文（TUC）的缩写不同，作为妥协，简称 UTC。

TIMESTAMP 与 DATETIME 除了存储字节和支持的范围不同外，还有一个最大的区别是：

DATETIME 在存储日期数据时，按实际输入的格式存储，即输入什么就存储什么，与时区无关；

而 TIMESTAMP 值的存储是以 UTC（世界标准时间）格式保存的，存储时对当前时区进行转换，检索时再转换回当前时区。即查询时，根据当前时区的不同，显示的时间值是不同的。

提示：如果为一个 DATETIME 或 TIMESTAMP 对象分配一个 DATE 值，结果值的时间部分被设置为'00:00:00'，因此 DATE 值未包含时间信息。如果为一个 DATE 对象分配一个 DATETIME 或 TIMESTAMP 值，结果值的时间部分被删除，因此 DATE 值未包含时间信息。

4.1.4 字符串类型

字符串类型用来存储字符串数据，还可以存储图片和声音的二进制数据。字符串可以区分或者不区分大小写的串比较，还可以进行正则表达式的匹配查找。

MySQL 中的字符串类型有 CHAR、VARCHAR、TINYTEXT、TEXT、MEDIUMTEXT、LONGTEXT、ENUM、SET 等。

下表中列出了 MySQL 中的字符串数据类型，括号中的 M 表示可以为其指定长度。

表 4-5 字符串类型

类型名称	说明	存储需求
CHAR(M)	固定长度非二进制字符串	M 字节，1<=M<=255
VARCHAR(M)	变长非二进制字符串	L+1 字节，在此，L<=M 和 1<=M<=255
TINYTEXT	非常小的非二进制字符串	L+1 字节，在此，L<2^8
TEXT	小的非二进制字符串	L+2 字节，在此，L<2^16
MEDIUMTEXT	中等大小的非二进制字符串	L+3 字节，在此，L<2^24
LONGTEXT	大的非二进制字符串	L+4 字节，在此，L<2^32
ENUM	枚举类型，只能有一个枚举字符串值	1 或 2 个字节，取决于枚举值的数目（最大值为 65535）
SET	一个设置，字符串对象可以有零个或多个 SET 成员	1、2、3、4 或 8 个字节，取决于集合成员的数量（最多 64 个成员）

VARCHAR 和 TEXT 类型是变长类型，其存储需求取决于列值的实际长度（在前面的表格中用 L 表示），而不是取决于类型的最大可能尺寸。

例如，一个 VARCHAR(10)列能保存一个最大长度为 10 个字符的字符串，实际的存储需要字符串的长度 L 加上一个字节以记录字符串的长度。对于字符“abcd”，L 是 4，而存储要求 5 个字节。

- CHAR 和 VARCHAR 类型

CHAR(M)为固定长度字符串，在定义时指定字符串列长。当保存时，在右侧填充空格以达到指定的长度。M 表示列的长度，范围是 0～255 个字符。

例如，CHAR(4)定义了一个固定长度的字符串列，包含的字符个数最大为 4。当检索到 CHAR 值时，尾部的空格将被删除。

VARCHAR(M)是长度可变的字符串，M 表示最大列的长度，M 的范围是 0～65535。VARCHAR 的最大实际长度由最长的行的大小和使用的字符集确定，而实际占用的空间为字符串的实际长度加 1。

例如，VARCHAR(50)定义了一个最大长度为 50 的字符串，如果插入的字符串只有 10 个字符，则实际存储的字符串为 10 个字符和 1 个字符串结束字符。VARCHAR 在值保存和检索时尾部的空格仍保留。

下面将不同的字符串保存到 CHAR(4)和 VARCHAR(4)列，说明 CHAR 和 VARCHAR 之间的差别，'␣'表示一个空格。如下表所示。

表 4-6 CHAR 型和 VARCHAR 型的比较

插入值	CHAR(4)	存储需求	VARCHAR(4)	存储需求
'␣'	'␣␣␣␣'	4 字节	''	1 字节
'ab'	'ab␣␣'	4 字节	'ab'	3 字节
'abc'	'abc␣'	4 字节	'abc'	4 字节
'abcd'	'abcd'	4 字节	'abcd'	5 字节
'abcdef'	'abcd'	4 字节	'abcd'	5 字节

对比结果可以看到，CHAR(4)定义了固定长度为 4 的列，无论存入的数据长度为多少，所占用的空间均为 4 个字节。VARCHAR(4)定义的列所占的字节数为实际长度加 1。

- TEXT 类型

TEXT 列保存非二进制字符串，如文章内容、评论等。当保存或查询 TEXT 列的值时，不删除尾部空格。TEXT 类型分为 4 种：TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT。不同的 TEXT 类型的存储空间和数据长度不同。

- TINYTEXT 表示长度为 255 字符的 TEXT 列。
- TEXT 表示长度为 65535 字符的 TEXT 列。
- MEDIUMTEXT 表示长度为 16777215 字符的 TEXT 列。
- LONGTEXT 表示长度为 4294967295 或 4GB 字符的 TEXT 列。

- ENUM 类型

ENUM 是一个字符串对象，值为表创建时列规定中枚举的一列值。其语法格式如下：

<字段名> ENUM('值 1', '值 1', …, '值 n')

字段名指将要定义的字段，值 n 指枚举列表中第 n 个值。

ENUM 类型的字段在取值时，能在指定的枚举列表中获取，而且一次只能取一个。如果创建的成员中有空格，尾部的空格将自动被删除。

ENUM 值在内部用整数表示，每个枚举值均有一个索引值；列表值所允许的成员值从 1 开始编号，MySQL 存储的就是这个索引编号，枚举最多可以有 65535 个元素。

例如，定义 ENUM 类型的列（'first'，'second'，'third'），该列可以取的值和每个值的索引如下表所示。

表 4-7 ENUM 类型

值	索引
NULL	NULL

''	0
'first	1
second	2
third	3

ENUM 值依照列索引顺序排列，并且空字符串排在非空字符串前，NULL 值排在其他所有枚举值前。

提示：ENUM 列总有一个默认值。如果将 ENUM 列声明为 NULL，NULL 值则为该列的一个有效值，并且默认值为 NULL。如果 ENUM 列被声明为 NOT NULL，其默认值为允许的值列表的第 1 个元素。

- SET 类型

SET 是一个字符串的对象，可以有零或多个值，SET 列最多可以有 64 个成员，值为表创建时规定的一列值。指定包括多个 SET 成员的 SET 列值时，各成员之间用逗号“,”隔开，语法格式如下：

SET('值 1', '值 2', …, '值 n')

与 ENUM 类型相同，SET 值在内部用整数表示，列表中每个值都有一个索引编号。当创建表时，SET 成员值的尾部空格将自动删除。

但与 ENUM 类型不同的是,ENUM 类型的字段只能从定义的列值中选择一个值插入,而 SET 类型的列可从定义的列值中选择多个字符的联合。

提示：如果插入 SET 字段中的列值有重复，则 MySQL 自动删除重复的值；插入 SET 字段的值的顺序并不重要，MySQL 会在存入数据库时，按照定义的顺序显示；如果插入了不正确的值，默认情况下，MySQL 将忽视这些值，给出警告。

4.1.5 二进制类型

MySQL 中的二进制字符串有 BIT、BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。

下表中列出了 MySQL 中的二进制数据类型，括号中的 M 表示可以为其指定长度。

表 4-8 二进制数据类型

类型名称	说明	存储需求
BIT(M)	位字段类型	大约 (M+7)/8 字节
BINARY(M)	固定长度二进制字符串	M 字节
VARBINARY (M)	可变长度二进制字符串	M+1 字节
TINYBLOB (M)	非常小的 BLOB	L+1 字节，在此，L<2^8
BLOB (M)	小 BLOB	L+2 字节，在此，L<2^16
MEDIUMBLOB (M)	中等大小的 BLOB	L+3 字节，在此，L<2^24
LONGBLOB (M)	非常大的 BLOB	L+4 字节，在此，L<2^32

- BIT 类型

位字段类型。M 表示每个值的位数，范围为 1～64。如果 M 被省略，默认值为 1。如果为 BIT(M)列分配的值的长度小于 M 位，在值的左边用 0 填充。例如，为 BIT(6)列分配一个值 b'101'，其效果与分配 b'000101'相同。

BIT 数据类型用来保存位字段值，例如以二进制的形式保存数据 13，13 的二进制形式为 1101，在这里需要位数至少为 4 位的 BIT 类型，即可以定义列类型为 BIT(4)。大于二进

制 1111 的数据是不能插入 BIT(4)类型的字段中的。

提示：默认情况下，MySQL 不可以插入超出该列允许范围的值，因而插入数据时要确保插入的值在指定的范围内。

- BINARY 和 VARBINARY 类型

BINARY 和 VARBINARY 类型类似于 CHAR 和 VARCHAR，不同的是它们包含二进制字节字符串。使用的语法格式如下：

列名称 BINARY(M) 或者 VARBINARY(M)

BINARY 类型的长度是固定的，指定长度后，不足最大长度的，将在它们右边填充 “\0” 补齐，以达到指定长度。例如，指定列数据类型为 BINARY(3)，当插入“a”时，存储的内容实际为“\a0\0”，当插入“ab”时，实际存储的内容为“ab\0”，无论存储的内容是否达到指定的长度，存储空间均为指定的值 M。

VARBINARY 类型的长度是可变的，指定好长度之后，长度可以在 0 到最大值之间。例如，指定列数据类型为 VARBINARY(20)，如果插入的值长度只有 10，则实际存储空间为 10 加 1，实际占用的空间为字符串的实际长度加 1。

- BLOB 类型

BLOB 是一个二进制的对象，用来存储可变数量的数据。BLOB 类型分为 4 种：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB，它们可容纳值的最大长度不同，如下表所示。

表 4-9 BLOB 类型

数据类型	存储范围
TINYBLOB	最大长度为 255 (28-1)字节
BLOB	最大长度为 65535 (216-1)字节
MEDIUMBLOB	最大长度为 16777215 (224-1)字节
LONGBLOB	最大长度为 4294967295 或 4GB (231-1)字节

BLOB 列存储的是二进制字符串（字节字符串），TEXT 列存储的是非进制字符串（字符字符串）。BLOB 列是字符集，并且排序和比较基于列值字节的数值；TEXT 列有一个字符集，并且根据字符集对值进行排序和比较。

4.1.6 MySQL 数据类型的选择

MySQL 提供了大量的数据类型，为了优化存储和提高数据库性能，在任何情况下都应该使用最精确的数据类型。

前面主要对 MySQL 中的数据类型及其基本特性进行了描述，包括它们能够存放的值的类型和占用空间等。本节主要讨论创建数据库表时如何选择数据类型。

可以说字符串类型是通用的数据类型，任何内容都可以保存在字符串中，数字和日期都可以表示成字符串形式。

但是也不能把所有的列都定义为字符串类型。对于数值类型，如果把它们设置为字符串类型的，会使用很多的空间。并且在这种情况下使用数值类型列来存储数字，比使用字符串类型更有效率。

另外需要注意的是，由于对数字和字符串的处理方式不同，查询结果也会存在差异。例如，对数字的排序与对字符串的排序是不一样的。

如果让 MySQL 把一个字符串列当作一个数字列来对待，会引发很严重的问题。这样做会迫使让列里的每一个值都执行从字符串到数字的转换，操作效率低。而且在计算过程中使用这样的列，会导致 MySQL 不会使用这些列上的任何索引，从而进一步降低查询的速度。

所以在选择数据类型时要考虑存储、查询和整体性能等方面的问题。

在选择数据类型时，首先要考虑这个列存放的值是什么类型的。一般来说，用数值类型列存储数字、用字符类型列存储字符串、用时态类型列存储日期和时间。

- 数值类型

对于数值类型列，如果要存储的数字是整数（没有小数部分），则使用整数类型；如果要存储的数字是小数(带有小数部分),则可以选用DECIMAL或浮点类型,但是一般选择FLOAT类型（浮点类型的一种）。

例如，如果列的取值范围是1~99999之间的整数，则MEDIUMINT UNSIGNED类型是最好的选择。

MEDIUMINT是整数类型，UNSIGNED用来将数字类型无符号化。比如INT类型的取值范围是-2147483648～2147483647，那么INT UNSIGNED类型的取值范围就是0～4294967295。

如果需要存储某些整数值，则值的范围决定了可选用的数据类型。如果取值范围是0~1000，那么可以选择SMALLINT~BIGINT之间的任何一种类型。如果取值范围超过了200万，则不能使用SMALLINT，可以选择的类型变为从MEDIUMINT到BIGINT之间的某一种。

当然，完全可以为要存储的值选择一种最“大”的数据类型。但是，如果正确选择数据类型，不仅可以使表的存储空间变小，也会提高性能。因为与较长的列相比，较短的列的处理速度更快。当读取较短的值时，所需的磁盘读写操作会更少，并且可以把更多的键值放入内存索引缓冲区里。

如果无法获知各种可能值的范围，则只能靠猜测，或者使用BIGINT以满足最坏情况的需要。如果猜测的类型偏小，那么也不是就无药可救。将来，还可以使用ALTER TABLE让该列变得更大些。

如果数值类型需要存储的数据为货币，如人民币。在计算时，使用到的值常带有元和分两个部分。它们看起来像是浮点值，但FLOAT和DOUBLE类型都存在四舍五入的误差问题，因此不太适合。因为人们对自己的金钱都很敏感，所以需要一个可以提供完美精度的数据类型。

可以把货币表示成DECIMAL(M,2)类型，其中M为所需取值范围的最大宽度。这种类型的数值可以精确到小数点后2位。DECIMAL的优点在于不存在舍入误差，计算是精确的。

对于电话号码、信用卡号和社会保险号都会使用非数字字符。因为空格和短划线不能直接存储到数字类型列里，除非去掉其中的非数字字符。但即使去掉了其中的非数字字符，也不能把它们存储成数值类型，以避免丢失开头的“零”。

- 日期和时间类型

MySQL对于不同种类的日期和时间都提供了数据类型，比如YEAR和TIME。如果只需要记录年份，则使用YEAR类型即可；如果只记录时间，可以使用TIME类型。

如果同时需要记录日期和时间，则可以使用TIMESTAMP或者DATETIME类型。由于TIMESTAMP列的取值范围小于DATETIME的取值范围,因此存储较大的日期最好使用DATETIME。

TIMESTAMP也有一个DATETIME不具备的属性。默认情况下，当插入一条记录但并没有指定TIMESTAMP这个列值时，MySQL会把TIMESTAMP列设为当前的时间。因此当需要插入记录和当前时间时，使用TIMESTAMP是方便的，另外TIMESTAMP在空间上比DATETIME更有效。

MySQL没有提供时间部分为可选的日期类型。DATE没有时间部分，DATETIME必须有时间部分。如果时间部分是可选的，那么可以使用DATE列来记录日期，再用一个单独的TIME列来记录时间。然后，设置TIME列可以为NULL。SQL语句如下：

```
CREATE TABLE mytb1 (
    date DATE NOT NULL,  #日期是必需的
    time TIME NULL  #时间可选(可能为NULL)
```

);

- 字符串类型

字符串类型没有像数字类型列那样的“取值范围”，但它们都有长度的概念。如果需要存储的字符串短于 256 个字符，那么可以使用 CHAR、VARCHAR 或 TINYTEXT。如果需要存储更长一点的字符串，则可以选用 VARCHAR 或某种更长的 TEXT 类型。

如果某个字符串列用于表示某种固定集合的值，那么可以考虑使用数据类型 ENUM 或 SET。

CHAR 和 VARCHAR 的区别如下:

CHAR 是固定长度字符，VARCHAR 是可变长度字符。

CHAR 会自动删除插入数据的尾部空格，VARCHAR 不会删除尾部空格。

CHAR 是固定长度，所以它的处理速度比 VARCHAR 的速度要快，但是它的缺点就是浪费存储空间。所以对存储不大，但在速度上有要求的可以使用 CHAR 类型，反之可以使用 VARCHAR 类型来实现。

存储引擎对于选择 CHAR 和 VARCHAR 的影响:

对于 MyISAM 存储引擎，最好使用固定长度的数据列代替可变长度的数据列。这样可以使整个表静态化，从而使数据检索更快，用空间换时间。

对于 InnoDB 存储引擎，最好使用可变长度的数据列，因为 InnoDB 数据表的存储格式不分固定长度和可变长度，因此使用 CHAR 不一定比使用 VARCHAR 更好，但由于 VARCHAR 是按照实际的长度存储，比较节省空间，所以对磁盘 I/O 和数据存储总量比较好。

- ENUM 和 SET

ENUM 只能取单值，它的数据列表是一个枚举集合。它的合法取值列表最多允许有 65535 个成员。因此，在需要从多个值中选取一个时，可以使用 ENUM。比如，性别字段适合定义，为 ENUM 类型，每次只能从‘男’或‘女’中取一个值。

SET 可取多值。它的合法取值列表最多允许有 64 个成员。空字符串也是一个合法的 SET 值。在需要取多个值的时候，适合使用 SET 类型，比如，要存储一个人兴趣爱好，最好使用 SET 类型。

ENUM 和 SET 的值是以字符串形式出现的，但在内部，MySQL 以数值的形式存储它们。

- 二进制类型

BLOB 是二进制字符串，TEXT 是非二进制字符串，两者均可存放大容量的信息。BLOB 主要存储图片、音频信息等，而 TEXT 只能存储纯文本文件。