03 Schema 设计规范是什么样的？

更新时间：2020-03-18 10:18:13

> 书是人类进步的阶梯。——高尔基

通常，我们所说的 Schema 设计指的是对数据表的的设计，这里，我将它的概念扩展为对库和表的设计。毕竟，我们在对 MySQL 基本的使用中，也就是根据业务需求去创建数据库、创建数据表。另外，我还是要做出一点强调：规范是一种通用的建议，并不一定适用于所有的场景，一定要仔细分析需求再做出合适的取舍。

1 Schema 设计的目标和原则

在讨论具体的设计规范之前，需要搞清楚我们追求的目标是什么，即遵循这样的规范，能够获得怎样的收益。同时，也要知道一些 MySQL 的基本特性，以此来把握设计的原则。

1.1 Schema 设计的目标

通俗的说，不论是 MySQL 还是其他工具也好，最基本的设计目标肯定是可用。如果可以，就在可用的基础之上，再去追求好用。下面，我来详细的对可用和好用的设计目标进行解读。

可用的设计目标

如果你设计的数据库和数据表能够支撑当前的业务需求，且在技术实现上没有太大的弊端，那么，我们就可以说它是可用的。更深层的看，这个设计目标的核心其实是对需求的理解。确实，理清了需求，你会得出结论：应该存储哪些数据、这些数据是什么类型、在代码中怎样使用这些数据等等。余下的建库建表也自然就是水到渠成了。

好用的设计目标

需求也许不会变化，但是随着业务量的增长触发数据和并发的增长，数据库是否还能保持相对较高的性能是个值得思考的问题，同时也是衡量设计目标是否好用的重要指标。

无论什么时候，我们对 MySQL（数据库）的使用都肯定是围绕数据的增删改查。而这些基本的操作，当数据量加速膨胀的过程中，也会引起性能瓶颈。所以，好用的设计目标讲究能够 “预见未来”，能够对未来做出预判。例如：将通用信息单独使用一张表存储、建立适当的索引等等。

1.2 Schema 设计的原则

鉴于 MySQL 的一些固有属性（特性），在使用上我们通常都会遵守一些 “共识”，而这些是与具体的业务没有相关性的。下面，我将会介绍一些通用的设计原则，它们有些是关于库的，有些是关于表的。

- 使用小写的名称，且只有英文字母：不论是库、表还是数据列，应该是只包含英文字母的名称，不要出现特殊字符或者是数字。这也比较好理解，英文字母不论是阅读还是编码都非常的便捷。另外，由于 MySQL 是大小写不敏感的，选择一律小写的名称能够统一书写规则，避免不必要的书写错误。
- 取一个有意义的名称，单词之间使用下划线连接：除了基本的名称书写规范之外，取一个有意义的名称是非常有必要的。例如：我们需要创建学生表，表的名称叫做 student 就会比 other 更易理解。当然，可能有些时候我们无法用一个单词表达清楚想要的含义，此时，可以使用多个单词，且单词之间使用下划线连接，例如：insert_time。最后，名称不要过长，最长不要超过 32 个字符。
- 记住 “够用且尽量小” 的原则：很明显，这条原则对应的是数据表列的数据类型选择问题。占用空间少的数据类型最直接的优势就是减少了用户数据存储空间和索引存储空间，这对于数据传输与检索的性能提高有着巨大意义。
- 不要使用物理外键：物理外键是说让数据库去管理表与表之间的关联关系，而它相对的逻辑外键，则是我们自己用代码去管理这种关系。这是因为物理外键存在两个重大缺陷：消耗数据库资源，降低数据库实例可扩展性；母表一旦受损，子表很难恢复，造成数据丢失。
- 表一定要有主键：MySQL 并不要求表一定要有主键，但是主键的作用是能够唯一区分表中的每一行。没有主键，更新或删除表中的特定行将会很困难，因为没有安全的方法保证只涉及相关的行。并且，主键能够为方便扩展、高可用的数据库系统做铺垫。
- 保持一致的字符集：库、表、数据列的字符集都应该是一致的，统一为 utf8 或 utf8mb4。字符集编码不仅影响数据存储，还会影响客户端与数据库之间的交互，最常见的问题就是字符集导致的乱码。所以，相同的字符集更利于管理，也更方便去排查问题。

2 库设计规范

MySQL 中库的概念是比较简单的，通常，我们会通过这样的语句去创建库：

```
CREATE DATABASE [IF NOT EXISTS] database_name;
```

库创建完了（符合之前所讲解的原则：名称、字符集等）之后，下一步就是在库中创建表。那么，一个库中允许存放多少个表呢 ？或者说，一个库中存储多少张表才合理呢 ？

2.1 一个库应该存储多少张表？

首先，我们可以使用如下的 SQL 语句查看系统库 mysql 中定义了多少张表：

```
mysql> SELECT COUNT(*) TABLES, table_schema FROM information_schema.TABLES WHERE table_schema = 'mysql' GROUP BY table_schema
;
+--------+--------------+
| TABLES | table_schema |
+--------+--------------+
|     31 | mysql        |
+--------+--------------+
```

可以看到，mysql 库（注意区分是系统库）中定义了 31 张表。由于它是 MySQL 的系统库，所以我们可以理所应当的认为它是合理的。换句话说，一个库中至少是可以存储 31 张表的。但是，得出这样的结论仅仅是靠猜测，没有任何站得住脚的依据。

MySQL 自身并没有对库的容量做出限制，也就是说，你几乎不用考虑表的数量上限问题。但是，当表的数量越多，越容易产生以下问题（以下所讨论的都是单个库）：

- 表越多，需要维护的元数据（表结构、统计信息等）就会越多。即使是这些元数据只占据很少的空间，但是也会让管理这些元数据变得很复杂，且通常也是不合理的需求分析造成的；
- 表越多，可能存储的数据量也会越大，这无疑会给数据库造成压力。且大量的数据聚集在同一个库中也是非常危险的，一旦出现库损坏，丢失的数据量也会更多。

综上所述，我们讨论了单库中表太多的缺陷，再去结合日常的工作实践来说，建议大家在一个库中创建的表数量不要超过 200。更常见的情况是，一个库中只维护几十到 100 张表。

3 表设计规范

不同于库比较单一的属性，在设计表时，需要考虑很多问题，毕竟我们操作的直接对象是表。下面，我将会分析讲解设计表时需要考虑的三点问题。

3.1 怎样理解范式和反范式

相信我们在刚开始接触数据库的时候就听过范式的概念，它的核心思想是数据只出现一次，不存在信息冗余。而反范式的概念也就是破坏了范式的规范，它允许出现冗余的数据。所以，对于这个问题来说，它聚焦的点在于：冗余字段是否是可取的。

对于任何给定的数据来说，我们可以设计各种各样的存储方案，从完全范式化到完全反范式化，或者兼顾两者。对于范式化的设计，有着这样的优点：

- 使用更少的存储空间
- 由于没有冗余存储，增删改查的速度相对较快

但是，如果我们想要的数据出现在两张或者多张表中，对于范式不存在冗余的设计，就不得不采用关联查询。而这恰恰是反范式设计最大的优势，适当的冗余设计，可以减少或避免表关联，提高查询效率。

所以，没有冗余就未必是好的，有时为了提高工作效率（对于查询大于更新的业务），就必须采用反范式的设计，适当的让数据存在冗余。

3.2 宽表与窄表怎样做出选择

字面意思理解，宽表就是数据列比较多的表，而窄表刚好相反，数据列比较少。MySQL 对于每张表有 4096 个列的硬限制，而真正在使用上的限制又会取决于你所使用的存储引擎。例如：对于 InnoDB 来说，一张表最多可以有 1017 列。在不考虑“宽和窄”的问题上来说，MySQL 和存储引擎支持的列数目肯定是足够的了。

在讲解宽表和窄表的优缺点之前，我这里给出一个定义：以 40 列为界，超过 40 列的表，我们可以称之为宽表，相对的，少于 40 列的表，我们称之为窄表。但同时，需要知道，这里的数字是人为定义的，MySQL 规范中并没有这种定义。我这里的划分是基于工作经验和总结，当然，你也可以有自己的数字界限。那么，不论是宽表还是窄表，它们一定各自都会有相应的优缺点：

- 窄表较多，数据列会更加分散，编写关联查询的难度就会很大
- 数据项会有不同的安全级别，宽表中涉及的列过多，数据权限的管理会带来很大的挑战
- 窄表数据量通常较少，但是等量的数据项会创建更多的表，管理难度大
- 宽表数据量通常较大，单表占据的存储空间过大，会降低排序、分组等查询的性能

综上所述，我们应该从多个角度去分析问题，完美的选择几乎是不存在的，我们在拥抱有利之处的时候，也不可避免的会摄入弊端，也正所谓鱼和熊掌不可兼得。

3.3 合理的索引是提升性能的关键

我们对索引的概念一定不会陌生，它能够加速表数据的查询，但是相应的，它也会占据一定的存储空间，也就是典型的以空间换时间的优化策略。另外，索引的存在，也会使插入、删除、更新的性能降低，因为这些操作都会伴随着索引的修改。所以，这一条设计规范所要追求的是空间与时间的平衡，达到既不占用过多的存储空间，也有较高的查询性能。

索引要建的合理，就必须要知晓并理解 MySQL 中索引创建和使用的特性：

- 一定要为作为搜索条件的字段创建索引，不是搜索条件的字段建索引反而会降低使用性能
- 选择区分度高的字段作为索引字段，重复性高的字段不要加索引
- 联合索引存在“最左前缀”的特性，不要建多余的索引

最后，如果一张表中已经存在了大量的数据，再去创建索引的过程会相当漫长，且可能会影响线上服务。此时，应该评估是否是在原表上增加索引还是创建新表并迁移数据。

4 总结

对于 Schema 设计规范，MySQL 并没有提供太多要求或建议，所以，更多的是经验之谈。我们在做工程，做设计的时候，不要总想着“一步到位”，过程一定是逐步迭代出来的。只有当你知道了这样的设计不好，你才能真正明白那样的设计是好的。多练习、多尝试、多总结，你也可以形成自己的一套规范。

5 问题

如果你的表没有定义主键，你知道 MySQL 会怎么做吗 ？

我需要创建一个学校库，存储教师、学生、院系等信息，你觉得叫做 school 好吗 ？ 为什么 ？

我编写的 SQL 语句需要做多表的 join 操作，应该给哪些列建索引呢 ？

6 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档

}

02 常用数据类型与使用建议

04 学会聚合与分组聚合是很有必要的