**数据库设计规范**

**郭国钊**

变更记录：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变更者 | 版本 | 变更日期 | 批准人 | 审批日期 | 变更内容 |
| 郭国钊 | V1.0 | 2017-06-12 |  |  | 创建 |
|  |  |  |  |  |  |

目录

[1 表设计规范 1](#_Toc486255231)

[2 索引设计规范 2](#_Toc486255232)

[3 SQL编写规范 2](#_Toc486255233)

[4 其他规范 3](#_Toc486255234)

# 表设计规范

* 尽量遵守数据库设计第三范式(3NF)。--未审计
* 第一范式（1NF）无重复的列
* 第二范式（2NF）属性完全依赖于主键
* 第三范式（3NF）属性不依赖于其它非主属性

例子：假定学生关系表为Student(学号, 姓名, 年龄, 所在学院, 学院地点, 学院电话)，关键字为单一关键字"学号"，因为存在如下决定关系：

(学号) → (姓名, 年龄, 所在学院, 学院地点, 学院电话)

这个数据库是符合2NF的，但是不符合3NF，因为存在如下决定关系：

(学号) → (所在学院) → (学院地点, 学院电话)

即存在非关键字段"学院地点"、"学院电话"对关键字段"学号"的传递函数依赖。

它也会存在数据冗余、更新异常、插入异常和删除异常的情况，读者可自行分析得知。

把学生关系表分为如下两个表：

学生：(学号, 姓名, 年龄, 所在学院)；

学院：(学院, 地点, 电话)。

这样的数据库表是符合第三范式的，消除了数据冗余、更新异常、插入异常和删除异常。

* 表名、列名一律采用英文命名，且必须有对应的中文注释(comment)！--审计
* 命名应使用常用英文词汇或者缩写，一律使用小写。多个单词组成的，以下划线(\_)分隔开来。--符合
* 名称中只能使用英文字母、数字和下划线，切勿使用空格。--符合
* 表、列名称中不允许使用数据库的关键字及保留字(如default、select)。--审计
* 命名长度不宜过长。若考虑可能的数据迁移兼容性，应尽量短小(如MySQL最长64字符，而Oracle中对象名称最大30个字符)。--符合
* 命名应该反映数据集所在的业务范围，或者业务功能，如property\_metric(资产度量)。--审计
* 当字段类型为枚举或者布尔型时，使用 ~~CHAR~~VARCHAR类型，长度取8或16，太长可考虑简写。常用的状态类字段~~注释可写成comment '定单状态：1-已下单，2-已付款，3-退款中，4-已退款'。~~使用大写的编码方式，如SMS-短信，PUSH-推送消息，INTERNAL-站内信等 --审计 讨论
* 在设计时候包含纯数据库功能字段，包括：type\_id，create\_time，create\_by，update\_time，update\_by，is\_deleted。这些字段与应用逻辑无关，记录的是数据行在数据库中的生命周期。特别是在按照时间进行增量数据迁移时，create\_time作用凸显！--审计
* 原则上每张表必须有主键，主键字段\*\_ID使用int或bigin。如果create语句中有类似AUTO\_INCREMENT=10000标示，则需去掉。--符合
* 常用于检索的字段(如姓名、手机)不允许为null，可提供空串’-’作为赋值。数字类型字段不允许为null，可提供0作为赋值。--审计
* 尽可能将长度较小的列放在前面，长度较大的列放在后面。--未审计讨论
* 每张表的列总数不宜过多，一般单表不应超过30列。若列过多，则应考虑垂直拆分。拆分的规则是将常用列(热列)放在主表，非常用列(冷列)放在辅表。--符合
* 若可预估表数据量很大，应预先考虑分库分表方案，即水平拆分。为防止分库分表后主键\*\_ID全局不唯一，对需分库分表的表使用UUID标识记录，以方便后期分库分表重构！--未审计
* 数据库默认字符编码设置为UTF8，默认存储引擎为INNODB。--符合
* 关于外键。数据模型设计外键，生成数据结构后disable外键。--未审计

# 索引设计规范

* WHERE条件后面的经常查询的字段，通常需要考虑创建索引。--审计
* 索引命名。普通索引，以idx\_开头，连接列名，如idx\_mobile。唯一索引，以unq\_开头，连接列名，如unq\_id\_card。--审计
* 值范围重复的比例少的列，适合作为建立索引字段。值范围重复比率高的字段不适合建立索引。--审计
* 主键字段自动创建唯一索引。--符合
* 慎用联合索引。某些情况下，联合索引可能失效，甚至可能导致执行计划异常。创建联合索引前，应经过反复的测试。--审计

# SQL编写规范

* 避免单SQL语句join过多数量表，尽可能将表总数控制在5张以下！--符合
* 应用程序中的SQL语句，禁止一切DDL、DCL操作，例如：create、drop、alter、grant、remove等。--符合
* 如果需要事务的支持，需使用innodb。程序建立数据库连接时，先关闭自动提交，如: set auto\_commit=0。 在写java代码的时候，在有事务处理的情况下，执行insert、delete、update后，再commit；在exception代码块，必须有相应的rollback操作。--未审计
* 不要出现select \*类似的代码，明确指定字段名，且仅选择必需的字段。--审计
* update、delete语句后面必须跟上where子句！--未审计 进行中
* 避免在where子句中对字段进行函数运算。如date\_format(p.payment\_date, '%Y-%m-%d') >= date\_format('2015-10-10', '%Y-%m-%d')，这样会使索引失效。--符合
* 避免多余的排序，使用group by的时候，默认会进行排序。若不需要排序，可以使用order by null。--未审计
* 当表连接时候，如果用于连接的两个表的字段数据类型不一致，则必须在一边加上类型转换的函数，以杜绝mysql做隐式类型转换。--未审计
* 关于绑定变量。对于OLAP/DSS类型的应用系统，可不使用绑定变量；对于OLTP类型的应用系统，在SQL语句中一定要使用绑定变量，能批量绑定更好；对于OLAP和OLTP混合型的应用系统，如果有循环，循环内部的SQL语句一定使用绑定变量。--未审计

# 其他规范

* 数据库应杜绝触发器。除非无其他解决办法，否则数据库中不应有触发器。这样既能有效保证数据库性能，又可以避免可能的触发器递归调用，那将会轻松导致系统崩溃！--符合
* 术业有专攻，数据库中避免创建存储过程、函数等子程序。避免将复杂、密集计算的业务逻辑封装进数据库，影响数据库性能，同时给应用程序的维护带来多层耦合。--符合
* 通常情况下，表设计应遵循第三范式。但在特定情况下，需要采取反范式手段，减少join，以冗余存储换取数据库性能。--未审计
* 视频、音频、图片、文档等大数据对象不宜存储在关系数据库中，可考虑存储在专门的文档数据库如MongoDB。--未审计 讨论