链滴

Solo 表结构

作者：88250

原文链接：https://ld246.com/article/1537694566175

来源网站：链滴

本文是《Solo 从设计到实现》的一个章节，该系列文章将介绍 Solo 这款 Java 博客系统是如何从无有的，希望大家能通过它对 Solo 从设计到实现有个直观地了解、能为想参与贡献的人介绍清楚项目也希望能为给~~重复发明~~重新定义博客系统的人做个参考
eart

本章我们主要介绍 Solo 的表结构定义。一些公共约定：

- 默认所有表都以 b3_solo_ 为表前缀
- 对于日期时间类型，我们使用的是 UTC 毫秒时间戳，这样主要是为了避免引入时区问题。统一使用 bigint 存储 UTC 时间戳，程序里面处理时区转换
- 没有外键约束
- 没有关联查询，关联查询场景通过多次查询实现

可以说 Solo 是以“轻数据库重程序”的思路来主导设计和实现的。

关联表

关联表的表结构非常类似，用于记录一种数据和另一种数据的关联。

列名	类型	长度	备注
oId	varchar	19	主键
type1_oId	varchar	19	型 1 的主键
type2_oId	varchar	19	型 2 的主键

在 Solo 中一共有 3 张关联表：

- archivedate_article：存档-文章关联
- category_tag：分类-标签关联
- tag_article：标签-文章关联

其实理论上可以将这 3 张表合并为一张，通过加入一个“类型”列来区别关联类型，并预留一些各种型的字段用于扩展。不过实际上我们并没有在 Solo 中这样做，因为表的数量并不重要，而且这样的象似乎意义不大。

不过既然都这样想了，我们在另一个博客系统中就真进行了尝试，有兴趣的话可以了解一下 Pipe。

用户表 b3_solo_user

列名	类型	长度	备注
oId	varchar	19	主键
userName	varchar	255	户名
userURL	varchar	255	户链接地址

userRole¹	varchar	55	用户角色
userAvatar	varchar	255	户头像图片地址
userB3Key	varchar	64	户 B3 Key
userGitHubId	varchar	32	户 GitHub User Id

- 1 用户角色的取值：
  - 管理员：adminRole
  - 普通用户：defaultRole
  - 访客用户：visitorRole

文章表 b3_solo_article

列名	类型	长度	备注
oId	varchar	19	主键
articleTitle	varchar	255	章标题
articleAbstract¹	text		章摘要
articleAbstractText	text		章摘要纯文本
articleTags²	text		章标签，英文逗号分隔
articleAuthorId	varchar	19	章作者 id
articleCommentCount	int	11	章评论计数
articleViewCount	int	11	章浏览计数
articleContent¹	mediumtext		章正文
articlePermalink	varchar	255	章访问路径
articlePutTop	char	1	章是否置顶
articleCreated	bigint	20	章创建时间戳
articleUpdated	bigint	20	章更新时间戳
articleRandomDouble³	double		

章随机数
articleSignId varchar 19
章签名档 id
articleCommentable char 1
章是否可评论
articleViewPwd⁴ varchar
00 文章浏览密码
articleImg1URL varchar 255
章首图地址
articleStatus int 11 文
状态⁵

- 1 摘要和正文都是存的 markdown 原文，因为 md 也支持直接用 HTML，所以存 HTML 也是没有题的，这主要看 markdown 渲染引擎的支持了
- 2 虽然有 tag_article 关联表，但这里也进行了标签存储，主要原因是 tag_article 关联表是为了展标签-文章列表，这里在 article 中单独存了 tag titles 是为了性能考虑，在渲染文章时不用多查（或者oin）一次关联表
- 3 文章随机数是用来实现随机文章的，MySQL 和 H2 虽然都提供了 RAND() 函数，但是性能实在差。通过查询时指定一个随机数作为条件和该字段比较就可以非常简单高效地查询随机文章列表
- 4 文章浏览密码如果留空表示不需要浏览密码，文章是公开可访问的
- 5 文章状态
  - 已发布：0
  - 草稿：1

评论表 b3_solo_comment

列名	类型	长度	备注
oId	varchar	19	主键
commentContent	text		
论内容			
commentCreated	bigint	20	
论创建时间戳			
commentName	varchar	50	
论人名称			
commentOnId	varchar	19	
论文章或页面的 id			
commentSharpURL	varchar	255	
论访问路径，带 # 锚点			
commentThumbnailURL	varchar	255	
论人头像地址			
commentURL	varchar	255	
论人链接			
commentOriginalCommentId¹	varchar		

9		父评论 id	
commentOriginalCommentName	varchar	50	评论人名称

- 1 评论因为可以回复，所以是个树形结构。用关系型数据库表示即需要一个 parent id 来保存父子系

导航表 b3_solo_page

列名	类型	长度	备注
oId	varchar	19	主键
pageOrder	int	11	页面航栏排序
pagePermalink	varchar	255	面访问路径
pageTitle	varchar	255	面标题
pageOpenTarget	varchar	255	面打开方式¹
pageIcon	varchar	255	面小图标地址

- 1 打开方式指的是 a 标签里的 target 属性值

标签表 b3_solo_tag

列名	类型	长度	备注
oId	varchar	19	主键
tagTitle	varchar	255	标标题

分类表 b3_solo_category

列名	类型	长度	备注
oId	varchar	19	主键
categoryTitle	varchar	64	类标题
categoryURI	varchar	255	类访问路径
categoryDescription	text		类描述
categoryOrder	int		分展现的排序
categoryTagCnt	int		分

下聚合的标签计数

文章分类是通过聚合标签实现的：

1. 将一组标签归类到一个分类下
2. 用户发布文章时使用这组标签中的一个或多个
3. 这篇文章将被自动聚合到该分类下
4. 一篇文章可被归属于多个分类下

这样设计能让分类成为动态的，文章不用固定设置分类，可以随时通过调整分类包含的标签来达到动聚合分类的效果。

存档日期表 b3_solo_archivedate

列名	类型	长度	备注
oId	varchar	19	主键
archiveTime	bigint	20	档日期时间，该月份第一天的时间戳

友链表 b3_solo_link

列名	类型	长度	备注
oId	varchar	19	主键
linkAddress	varchar	255	接地址
linkDescription	varchar	255	接描述
linkOrder	int	11	链接展的排序
linkTitle	varchar	255	接标题

支持友链这个特性其实颇为画蛇添足，因为完全可以用自定义页面来实现友链页面。更进一步，自定页面其实也是比较累赘的，其实只需要自定义导航，然后将这个导航跳转到某篇文章链接即可。这些度设计在 Pipe 中得到了改进。

插件表 b3_solo_plugin

列名	类型	长度	备注
oId	varchar	19	主键
author	text¹		件作者
name	varchar	255	插件称

status	varchar	10	插件态²
version	varchar	10	插版本
setting	text		插件数据，SON³

- 1 插件作者列值允许使用 HTML，所以这个字段用了 text 类型
- 2 插件状态
  - 启用：ENABLED
  - 禁用：DISABLED
- 3 插件相关的数据以 JSON 字符串形式保存

配置表 b3_solo_option

列名	类型	长度	备注
oId¹	varchar	19	键
optionValue	text		配置值
optionCategory	varchar	20	置项分类

- 1 主键没有用时间戳，而是配置项键名，在 Option.java 中的 OPTION_ID_C_* 定义

表设计小节

Solo 的数据库设计我们一直在进行着修正和完善，相信最终我们会让它达到一个简约的状态，让大能够一目了然的同时不影响功能和性能。

当然，这不仅仅是数据库设计的目标，在整个项目的技术方面，我们都会尽量朝着这个目标努力，好并**可读**的程序才有可能成为一个受欢迎的项目，才有可能让更多的开源爱好者参与进来。