



数据库设计和使用规范

编写	张思宇	日期	2017-02-16
审核		日期	
批准		日期	

一配网络科技（北京）有限公司中心

（内部资料，注意保密）

修改记录

版本	日期	修改纪要	修改人
V0.1	2017-02-16	创建	张思宇

目 录

目 录.....	2
1. 概述.....	1
1.1 简介.....	1
1.2 适用范围.....	1
1.3 缩写和术语.....	1
1.4 参考资料.....	1
2. 基础规范.....	1
3. 命名规范.....	1
4. 表设计规范.....	2
5. 字段设计规范.....	3
6. 索引设计规范.....	3
7. SQL使用规范.....	4
8. 附录.....	5

1. 概述

1.1 简介

本文的目的是制定数据库设计和使用规范，在大家写后台数据库时根据该规范来操作。

1.2 适用范围

后端数据库设计规范，后端数据库使用与开发人员应遵守该文档

1.3 缩写和术语

无

1.4 参考资料

[1] 1pei-IT 2015-0014.9-云进销存需求设计说明书V2.9

[2] 58到家数据库30条军规解读，2017.02.16

2. 基础规范

(1) 必须使用InnoDB存储引擎

解读：支持事务、行级锁、并发性能更好、CPU及内存缓存页优化使得资源利用率更高

(2) 必须使用UTF8字符集

解读：万国码，无需转码，无乱码风险，节省空间

(3) 数据表、数据字段必须加入中文注释

解读：N年后谁知道这个r1,r2,r3字段是干嘛的

(4) 禁止使用存储过程、视图、触发器、Event

解读：高并发大数据的互联网业务，架构设计思路是“解放数据库CPU，将计算转移到服务层”，并发量大的情况下，这些功能很可能将数据库拖死，业务逻辑放到服务层具备更好的扩展性，能够轻易实现“增机器就加性能”。数据库擅长存储与索引，CPU计算还是上移吧

(5) 禁止存储大文件或者大照片

解读：为何要让数据库做它不擅长的事情？大文件和照片存储在文件系统，数据库里存URI多好

3. 命名规范

(6) 只允许使用内网域名，而不是ip连接数据库

(7) 线上环境、开发环境、测试环境数据库内网域名遵循命名规范

业务名称: xxx

线上环境: 1pei.xxx.db

开发环境: 1pei.xxx.rdb

测试环境: 1pei.xxx.tdb

从库在名称后加-s标识, 备库在名称后加-ss标识

线上从库: 1pei.xxx-s.db

线上备库: 1pei.xxx-sss.db

(8) 库名、表名、字段名: 小写, 下划线风格, 不超过32个字符, 必须见名知意, 禁止拼音英文混用

(9) 表名t_XXX, 非唯一索引名idx_XXX, 唯一索引名uniq_XXX

4. 表设计规范

(10) 单实例表数目必须小于500

(11) 单表列数目必须小于30

(12) 表必须有主键, 例如自增主键

解读:

a) 主键递增, 数据行写入可以提高插入性能, 可以避免page分裂, 减少表碎片提升空间和内存的使用

b) 主键要选择较短的数据类型, Innodb引擎普通索引都会保存主键的值, 较短的数据类型可以有效减少索引的磁盘空间, 提高索引的缓存效率

c) 无主键的表删除, 在row模式的主从架构, 会导致备库夯住

(13) 禁止使用外键, 如果有外键完整性约束, 需要应用程序控制

解读: 外键会导致表与表之间耦合, update与delete操作都会涉及相关联的表, 十分影响sql的性能, 甚至会造成死锁。高并发情况下容易造成数据库性能, 大数据高并发业务场景数据库使用以性能优先

5. 字段设计规范

(14) 必须把字段定义为NOT NULL并且提供默认值

解读:

a) null的列使索引/索引统计/值比较都更加复杂,对MySQL来说更难优化

b) null这种类型MySQL内部需要进行特殊处理,增加数据库处理记录的复杂性;同等条件下,表中有较多空字段的时候,数据库的处理性能会降低很多

c) null值需要更多的存储空间,无论是表还是索引中每行中的null的列都需要额外的空间来标识

d) 对null的处理时候,只能采用is null或is not null,而不能采用=、in、<、<>、!=、not in这些操作符号。如: where name!= 'shenjian', 如果存在name为null值的记录,查询结果就不会包含name为null值的记录

(15) 禁止使用TEXT、BLOB类型

解读: 会浪费更多的磁盘和内存空间,非必要的大量的大字段查询会淘汰掉热数据,导致内存命中率急剧降低,影响数据库性能

(16) 禁止使用float或double数据类型存储货币

解读: 使用decimal, 小数数据类型容易导致钱对不上

(17) 必须使用varchar(20)存储手机号

解读:

a) 涉及到区号或者国家代号,可能出现+(-)

b) 手机号会去做数学运算么?

c) varchar可以支持模糊查询,例如: like "138%"

(18) 禁止使用ENUM,可使用TINYINT代替

解读:

a) 增加新的ENUM值要做DDL操作

b) ENUM的内部实际存储就是整数,你以为自己定义的是字符串?

6. 索引设计规范

(19) 单表索引建议控制在5个以内

(20) 单索引字段数不允许超过5个

解读：字段超过5个时，实际已经起不到有效过滤数据的作用了

(21) 禁止在更新十分频繁、区分度不高的属性上建立索引

解读：

a) 更新会变更B+树，更新频繁的字段建立索引会大大降低数据库性能

b) “性别”这种区分度不大的属性，建立索引是没有什么意义的，不能有效过滤数据，性能与全表扫描类似

(22) 建立组合索引，必须把区分度高的字段放在前面

解读：能够更加有效的过滤数据

7. SQL使用规范

(23) 禁止使用SELECT *，只获取必要的字段，需要显示说明列属性

解读：

a) 读取不需要的列会增加CPU、IO、NET消耗

b) 不能有效的利用覆盖索引

c) 使用SELECT *容易在增加或者删除字段后出现程序BUG

(24) 禁止使用INSERT INTO t_xxx VALUES(xxx)，必须显示指定插入的列属性

解读：容易在增加或者删除字段后出现程序BUG

(25) 禁止使用属性隐式转换

解读：SELECT uid FROM t_user WHERE phone=13812345678 会导致全表扫描，而不能命中phone索引，猜猜为什么？（这个线上问题不止出现过一次）

(26) 禁止在WHERE条件的属性上使用函数或者表达式

解读：SELECT uid FROM t_user WHERE from_unixtime(day)>='2017-02-15' 会导致全表扫描

正确的写法是：SELECT uid FROM t_user WHERE day>= unix_timestamp('2017-02-15 00:00:00')

(27) 禁止负向查询，以及%开头的模糊查询

解读：

a) 负向查询条件：NOT、!=、<>、!<、!>、NOT IN、NOT LIKE等，会导致全表扫描

b) %开头的模糊查询，会导致全表扫描

(28) 禁止大表使用JOIN查询，禁止大表使用子查询

解读：会产生临时表，消耗较多内存与CPU，极大影响数据库性能

(29) 禁止使用OR条件，必须改为IN查询

解读：旧版本Mysql的OR查询是不能命中索引的，即使能命中索引，为何要让数据库耗费更多的CPU帮助实施查询优化呢？

(30) 应用程序必须捕获SQL异常，并有相应处理

总结：大数据量高并发的互联网业务，极大影响数据库性能的都不让用，不让用哟。

8. 附录

无