MySQL数据库设计规范

(2018-05-28, created by Zhu Yin)

基础规范

- 表存储引擎必须使用InnoDB
- 表字符集默认使用utf8，必要时候使用utf8mb4

> 解读：
>
> 1. 通用，无乱码风险，汉字3字节，英文1字节；
> 2. utf8mb4是utf8的超集，有存储4字节例如表情符号时，使用它。

- 禁止使用存储过程，视图，触发器，Event

> 解读：
>
> 1. 对数据库性能影响较大，互联网业务，能让站点层和服务层干的事情，不要交到数据库层
> 2. 调试，排错，迁移都比较困难，扩展性较差

- 禁止在数据库中存储大文件，例如照片，可以将大文件存储在对象存储系统，数据库中存储路径
- 禁止在线上环境做数据库压力测试
- 测试，开发，线上数据库环境必须隔离
- 不在数据库做计算，cpu计算务必移至业务层
- 控制单表数据量，单表记录控制在千万级
- 控制列数量，字段数控制在20以内
- 平衡范式与冗余，为提高效率可以牺牲范式设计，冗余数据
- 拒绝3B(big)，大sql，大事务，大批量

命名规范

- 库名，表名，列名必须用小写，采用下划线分隔

> 解读：abc，Abc，ABC都是给自己埋坑

- 库名，表名，列名必须见名知义，长度不要超过32字符

> 解读：tmp，wushan谁TM知道这些库是干嘛的

- 库备份必须以bak为前缀，以日期为后缀
- 从库必须以-s为后缀
- 备库必须以-ss为后缀

表设计规范

- 单实例表个数必须控制在2000个以内
- 单表分表个数必须控制在1024个以内
- 表必须有主键，推荐使用UNSIGNED整数为主键

> 主键不应该被修改
>
> 字符串不应该做主键
>
> 如果不指定主键，innodb会使用唯一且非空值索引代替
>
> 潜在坑：删除无主键的表，如果是row模式的主从架构，从库会挂住

- 禁止使用外键，如果要保证完整性，应由应用程式实现

> 解读：外键使得表之间相互耦合，影响update/delete等SQL性能，有可能造成死锁，高并发情况下容易成为数据库瓶颈。

- 建议将大字段，访问频度低的字段拆分到单独的表中存储，分离冷热数据

列设计规范

- 根据业务区分使用tinyint/int/bigint，分别会占用1/4/8字节
- 根据业务区分使用char/varchar

> 解读：
>
> 1. 字段长度固定，或者长度近似的业务场景，适合使用char，能够减少碎片，查询性能高
> 2. 字段长度相差较大，或者更新较少的业务场景，适合使用varchar，能够减少空间

- 根据业务区分使用datetime/timestamp

> 解读：
>
> 1. datetime占用8个字节，日期范围是1001-9999年，与时区无关
> 2. timestamp占用4个字节，时间范围是1970-2038年，存储时，从当前时区转化为UTC（世界标准时间），即存储时间与时区有关，显示的值也依赖于时区，默认不为空（not null），默认值为当前时间（CURRENT_TIMESTAMP）
> 3. 存储年使用YEAR，存储日期使用DATE，存储时间使用datetime

- 必须把字段定义为NOT NULL并设默认值

> 解读：
>
> 1. NULL的列使用索引，索引统计，值都更加复杂，MySQL更难优化

> 2. NULL需要更多的存储空间
> 3. NULL只能采用IS NULL或者IS NOT NULL，而在=/!=/in/not in时有大坑

- 使用INT UNSIGNED存储IPv4，不要用char(15)
- 使用varchar(20)存储手机号，不要使用整数

> 解读：
>
> 1. 牵扯到国家代号，可能出现+/-/()等字符，例如+86
> 2. 手机号不会用来做数学运算
> 3. varchar可以模糊查询，例如like '138%'

- 使用TINYINT来代替ENUM

> 解读：ENUM增加新值要进行DDL操作

- 有些字符转化为数
  - 用int而不是char(15)存储ip

索引规范

- 唯一索引使用uniq_[字段名]来命名
- 非唯一索引使用idx_[字段名]来命名
- 单张表索引数量建议控制在5个以内

> 解读：
>
> 1. 互联网高并发业务，太多索引会影响写性能
> 2. 生成执行计划时，如果索引太多，会降低性能，并可能导致MySQL选择不到最优索引
> 3. 异常复杂的查询需求，可以选择ES等更为适合的方式存储*

- 组合索引字段数不建议超过5个

> 解读：如果5个字段还不能极大缩小row范围，八成是设计有问题

- 不建议在频繁更新的字段上建立索引
- 覆盖记录条数过多不适合建索引，例如"性别"
- 非必要不要进行JOIN查询，如果要进行JOIN查询，被JOIN的字段必须类型相同，并建立索引

> 解读：踩过因为JOIN字段类型不一致，而导致全表扫描的坑么？

- 理解组合索引最左前缀原则，避免重复建设索引，如果建立了(a,b,c)，相当于建立了(a), (a,b), (a,b,c)

SQL规范

- 禁止使用select *，只获取必要字段

> 解读：
>
> 1. select *会增加cpu/io/内存/带宽的消耗
> 2. 指定字段能有效利用索引覆盖
> 3. 指定字段查询，在表结构变更时，能保证对应用程序无影响

- insert必须指定字段，禁止使用insert into T values()

> 解读：指定字段插入，在表结构变更时，能保证对应用程序无影响

- 隐式类型转换会使索引失效，导致全表扫描
- 禁止在where条件列使用函数或者表达式

> 解读：导致不能命中索引，全表扫描
>
> 1. 例如：`select id where age +1 = 10;`

- 禁止负向查询以及%开头的模糊查询

> 解读：导致不能命中索引，全表扫描

- 禁止大表JOIN和子查询
- 同一个字段上的OR必须改写问IN，IN的值必须少于50个
- 应用程序必须捕获SQL异常

> 解读：方便定位线上问题

- sql语句尽可能简单
  - 一条sql只能在一个cpu运算
  - 大语句拆小语句，减少锁时间
  - 一条大sql可以堵死整个库
- limit高效分页 limit越大，效率越低
  - `select id from t limit 10000, 10;` 应该改为 =>`select id from t where id > 10000 limit 10;`
- 务必请使用“同类型”进行比较，否则可能全表扫面
- 使用新能分析工具
  - show profile;
  - mysqlsla;
  - mysqldumpslow;
  - explain;
  - show slow log;
  - show processlist;
  - show query_response_time(percona)

参考

- 58到家MySQL军规升级版
- 赶集mysql军规