MySQL 设计规范

SOHO4 课 程祺

版 本 历 史

版本/状态	责任人	起止日期	备注
V0.1/草稿	程祺	15June18	撰写文档。
V0.2/草稿	程祺	29June18	参考阿里和微软的规范修改文档
V0.3/草稿	程祺	3Jul18	修正部分内容，添加 SQL 关键字等信息
V0.4/草稿	程祺	6Jul18	按照评审结果补充修改文档

目 录

1. 数据库中表设计规范

1.1 数据库表字段定义规范

1) 【强制】小数类型为 decimal，禁止使用 float 和 double

说明:float 和 double 在存储的时候有精度损失问题，在进行筛选的时候可能存在无法正确匹配问题

2) 【强制】如果某一个字段的长度基本差不多，优先使用 char 而非 varchar

说明:char 在 MySQL 中的存储采用的是固定长度的块进行存储，如果实际长度不足，那么 char 类型的字段将在后面补充空格，检索的时候去掉空格。varchar 是长度加 value 的形式存储的，检索的时候空格不会被去除。从检索效率上来看 char 要比 varchar 来的高

3) 【强制】varchar 的存储范围虽然可以达到 65535，但是实际使用时不应该超过 5000，超过 5000 的字段应该独立成一张表并且使用 text 来存储，然后存储其索引

说明:MySQL 在很多检索的时候需要读取一整行记录，每行记录越大那么其 IO 消耗越多，会影响性能，所以存在某一列字段特别长的时候应该独立出来，这样不会影响主表的性能

4) 【强制】MySQL 的一行不能超过 65535（不包括 Blob），所以使用 varchar 的时候要注意字段长度

说明:超过 65535 后插入记录会失败

5) 【强制】所有独一无二的字段必须加 unique 限制，独一无二的判断由数据库完成

说明:unique 限制可以由上层逻辑实现，例如通过复杂的 SQL 语句或者锁表判断是否存在后再插入新的记录，也可以交给数据库来实现，交给数据库的好处就是简化了上层的代码逻辑并且效率更高

6) 【建议】数据表中建议保存 gmt_create, gmt_modified 两个字段。Create 字段表示创建时间 Modified 字段表示修改时间

说明:这两个字段作用类似于文件的创建和修改时间，如果有这两个字段需要手动维护

7) 【建议】数据库中表示整数类型的如果不会出现负数，则添加上 unsigned 属性

8) 【建议】数据库中所有不可能为 null 的字段添加 NOT NULL 限制

说明: 存在 NOT NULL 限制的话，上层代码如果出现 BUG 导致该字段变成 NULL，数据库会出错用于 debug 上层代码

9) 【建议】MySQL 中所有字段应该由默认值，尽量不要在表中的记录出现 NULL

说明:数据库记录的中的列如果有 NULL，对数据性能有一定影响。并且很多函数对 NULL 处理不友好，容易出错

10)【建议】对于数据库写入性能要求较高，并且有集群可能的时候不使用外键，外键的功能由上层代码控制。对于性能要求较低的小系统我们也尽量不要使用外键。如果使用上层代码完成外键的功能，在开发阶段应该添加 district 模式，用于 debug 上层代码

说明:外键是为了保证数据引用的一致性，会影响插入和更新，district 模式保证有子表引用的时候更新或者删除主表会出现删除或者修改失败，可以用于判断上层逻辑是否存在缺陷，***如果需要使用外键的时候需要评审通过***

1.2 索引使用的规范

1) 【强制】对于任何具有唯一特性的字段，包括复合键，应该建立唯一索引

说明:唯一特性建立索引可以极大的提高检索的效率，并且用户态的检察唯一性很容易出现 BUG

2) 【强制】禁止 3 个表以上进行 join 查询，多表联查的时候保证被关联的字段需要建立相关索引

说明:大量的表进行 join 联查会严重影响性能

3) 【强制】在 varchar 上面建立索引的时候必须指定索引的长度，建议长度为 20

说明:索引是需要存储空间的并且字段越长，建立索引需要的时间也越长，根据统计长度为 20 的索引区分度可以达到 90%

4) 【建议】如果存在左模糊或者全模糊搜索，考虑是否可以使用其它的方案代替

说明:索引文件具有 B-Tree 的最左前缀匹配，如果使用左模糊匹配将无法使用索引

5) 【建议】如果有 order by 的场景，注意使用索引的有序性

说明:如下两个例子说明如何使用索引

正例: where a=? and b=? order by c; 索引：a_b_c

反例: 索引中有范围查找的时候，那么索引将无效，例如 where a > 10 order by b

6) 【建议】利用延迟关联或者子查询优化超多分页的场景

说明:延迟关联的例子SELECT a.* FROM 表_1 a, (select id from 表1 where 条件 LIMIT 10000, 20) b where a.id = b.id

7) 【建议】建立索引的时候区分度最高的放在最左边

说明:如果第一个字段区分度非常高，基本是唯一的，那么后面区分度低的甚至不用建立索引，区分度高的放在左边有利于检索

8) 【建议】存在范围判断时，即使该字段的区分度高也要放到后面去

说明:如果存在非等号和等号联合进行查询例如select * where a.id == ? and a.c > ?,此时即使c的辨识度较高建立的索引也应该是idx_id_c

1.3 SQL 语句使用规范

1) 【强制】不要使用 count（列名）或者 count(常量)来替代 count(*), count(*)是 SQL92 定义的标准统计函数的语法，和数据库无关，跟 NULL 和非 NULL 无关

   说明: count(*)会统计值为 NULL 的行，而 count(列名)不会统计出现 NULL 的行

2) 【强制】count(distinct col)计算除 NULL 之外不重复的行数

3) 【强制】当某一列的值全是 NULL 时，count(col)的返回结果是 0，但 sum(col)的返回结果为 NULL，使用 sum 时注意 NPE 问题

4) 【强制】使用 ISNULL()来判断是否为 NULL 值

   说明: NULL与任何值的直接比较都为NULL

   1. NULL<>NULL的返回结果是NULL，而不是false

   2. NULL=NULL的返回结果是NULL，而不是true

   3. NULL<>1的返回结果是NULL，而不是true

5) 【强制】在书写分页查询时，若 count 为 0 直接返回，避免执行后面的分页语法

6) 【强制】禁止使用存储过程

   说明:存储过程是一组为了完成特定功能的 SQL 语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行它。这种方法难以调试和扩展，更没有移植性可言，***如果要使用存储过程一定要明白自己在做什么，并且组织评审通过后方可使用***

7) 【建议】避免使用 in 操作，如果真的要使用保证 in 的集合在 1000 个之内

   说明:当 in 的集合比较多的时候会有效率问题，除非能保证后面的 in 的集合不会变大，否则避免使用 in

1.4 数据库分库分表规范

1) 【建议】单表行数超过 500 万行或者单表的容量超过 2GB，才推荐进行分库分表

   说明:如果预计 3 年内的数据量根本达不到这个级别，请不要创建表时就分库分表

2) 【建议】垂直拆分只针对一行记录超过 8KB，并且其中有一些字段对于这条记录来说用的频率较低

   说明:MySQL 的 InodeDB 页的默认大小是 16KB（UNIV_PAGE_SIZE 宏控制），但是为了保存 B-Tree 结构，其保存数据会缩减到 9K 左右，所以这里我们定义一条记录应该低于 8KB

   这里我们暂定最大值超过 8K 即开始拆分(一般出现这种情况就意味着其中存在很大的 varchar 之类)

3) 【建议】水平拆分针对单表函数过大按照 1 中说明的超过 500 万行就可以考虑水平拆分

2. ORM 映射

1) 【强制】在表的查询中禁止使用*一次性获取全部属性，要做到那些字段使用查询那些字段

2) 【强制】sql.xml 等配置中使用:#{}, #param# 不要使用${}，后面一种方式容易出现 SQL 注入

3) 【强制】不要写一个很大的接口更新所有的数据，这样效率低，而且容易出错

4) 【强制】不允许直接拿 HashMap 与 Hashtable 作为查询结果集的输出

   说明：resultClass="Hashtable"，会置入字段名和属性值，但是值的类型不可控

5) 【建议】@Transactional 事务不要滥用。事务会影响数据库的 QPS，另外使用事务的地方需要考虑各方面的回滚方案，包括缓存回滚、搜索引擎回滚、消息补偿、统计修正等

3. MySQL 数据的基本类型


MySQL类型.xlsx

4. SQL 保留字


MySQL 5.7保留字.txt


Oracle数据库保留字.txt


SQL server保留字.txt

5. SQL 语句优化

1) 【强制】in 和 exist 在子查询中，如果子查询的表小用 in，子查询的表较大使用 exist

   说明:in 的查询方式为先进行子查询，然后对每一个子查询在主表里面查询，exist 正好对先便利第一个表，然后对第一个表的结果进行第二次存在性查询。这两个大原则就是让第一次查询的数据尽量少

2) 【建议】一般来说 join 的效率要比子查询要高，多表联查的时候优先考虑 join。小项目不用考虑效率问题，数据库不是很大的时候 join 和子查询不会有太大的差别

3) 【建议】引用自<高性能的 MySQL>,在大系统中对于级联的查询应该分解为不同的查询

   说明:分解为不同的查询有如下好处:

   1. 让缓存更高效
   2. 分解查询后可以降低所得竞争
   3. 应用层关联具有更强的扩展性
   4. MySQL 中使用的是循环关联，应用层可以使用更高效的关联方式

5.1 无线云管理性能优化中的建议

1) 【禁止】分库的情景下禁止使用级联操作

2) 【禁止】数据库读取尽量使用批量方式的度

3) 【禁止】数据库插入时采用 insert into values 一次写入多条数据，减少事物的数量

4) 【建议】批量修改多条记录为相同值时，采用 update set xxx where id in（）方式

5) 【建议】批量操作时，尽量减少事务的创建

5.2 无线云管理性能优化文档

无线云管理数据读写优化总结.doc

6. MySQL 缩写汇总表

MySQL缩写信息表.xlsx