MySQL规范

字段规范

- 库名、表名、字段名必须使用小写字母，“_”分割
- 库名、表名、字段名必须不超过12个字符
- 库名、表名、字段名见名知意,建议使用名词而不是动词
- 建议使用InnoDB存储引擎
- 建议使用utf8mb4字符集
- 存储精确浮点数必须使用DECIMAL替代FLOAT和DOUBLE
- 建议使用UNSIGNED存储非负数值
- 建议使用INT UNSIGNED存储IPV4
- 整形定义中不添加长度，比如使用INT，而不是INT(4)
- 使用短数据类型，比如取值范围为0-80时，使用TINYINT UNSIGNED
- 每个表需要建立id，created_time，modified_time

- 不建议使用ENUM类型，使用TINYINT来代替
- 尽可能不使用TEXT、BLOB类型
- VARCHAR(N)，N表示的是字符数不是字节数，比如VARCHAR(255)，可以最大可存储255个汉字，需要根据实际的宽度来选择N
- VARCHAR(N)，N尽可能小，因为MySQL一个表中所有的VARCHAR字段最大长度是65535个字节，进行排序和创建临时表一类的内存操作时，会使用N的长度申请内存
- 表字符集选择UTF8MB4
- 存储年使用YEAR类型
- 存储日期使用DATE类型
- 存储时间（精确到秒）建议使用datetime类型,datetime不受时区影响，存储范围较大，新版的mysql，datetime和timestramp存储空间都是4个字符
- 建议字段定义为NOT NULL
- 将过大字段拆分到其他表中
- 禁止在数据库中使用VARBINARY、BLOB存储图片、文件等

索引规范

- 非唯一索引必须按照“idx_字段名称_字段名称[_字段名]”进行命名
- 唯一索引必须按照“uniq_字段名称_字段名称[_字段名]”进行命名
- 索引名称必须使用小写
- 索引中的字段数建议不超过5个
- 单张表的索引数量控制在5个以内
- 唯一键由3个以下字段组成
- 数据表中一定要建立主键
- 索引字段的顺序需要考虑字段的查询频率和差异性，查询频率大的和差异性大的放在左边
- 使用ORDER BY，GROUP BY，DISTINCT的字段放在索引的后边，这些函数导致索引失效
- 使用EXPLAIN判断SQL语句是否合理使用索引，尽量避免extra列出现：Using File Sort，Using Temporary
- UPDATE、DELETE语句需要根据WHERE条件添加索引
- 不建议使用%前缀模糊查询，例如LIKE “%”
- 合理创建联合索引（避免冗余），(a,b,c) 相当于 (a) 、(a,b) 、(a,b,c)

sql语句

- 使用prepared statement（预处理），可以提供性能并且避免SQL注入
- SQL语句中IN包含的值不应过多
- UPDATE、DELETE语句不使用LIMIT
- WHERE条件中必须使用合适的类型，避免MySQL进行隐式类型转化
- SELECT语句只获取需要的字段
- SELECT、INSERT语句必须显式的指明字段名称，不使用SELECT *，不使用INSERT INTO table()
- WHERE条件中的非等值条件（IN、BETWEEN、<、<=、>、>=）会导致后面的条件字段使用不了索引
- 避免在SQL语句进行数学运算或者函数运算，容易将业务逻辑和DB耦合在一起

- INSERT语句使用batch提交（INSERT INTO table VALUES(),(),()……），values的个数不应过多
- 避免使用存储过程、触发器、函数等，容易将业务逻辑和DB耦合在一起，并且MySQL的存储过程、触发器、函数中存在一定的bug
- 避免使用JOIN
- 使用合理的SQL语句减少与数据库的交互次数
- 不使用ORDER BY RAND()，使用其他方法替换
- 建议使用合理的分页方式以提高分页的效率
- 统计表中记录数时使用COUNT(*)，而不是COUNT(primary_key)和COUNT(1)
- 禁止在从库上执行后台管理和统计类型功能的QUERY

FAQ

- 为什么使用InnoDB存储引擎?
- 支持事务，行级锁，更好的恢复性，高并发下性能更好，cpu及内存缓存页优化使得资源利用率更高
- 存储精确浮点数必须使用DECIMAL替代FLOAT和DOUBLE?
- decimal精度更高，计算机的浮点数内部都是二进制表示的，将一个十进制数转换为二进制浮点数时，会造成误差（131072.67，131072.68）
- 为什么使用UNSIGNED存储非负数值?
- 同样的字节数，存储的数值范围更大
- INT[M]，M值代表什么含义?
- 注意数值类型括号后面的数字只是表示宽度而跟存储范围没有关系，比如INT(3)默认显示3位，空格补齐，超出时正常显示
- 不建议使用ENUM、SET类型，使用TINYINT来代替?
- 添加新的值要做DDL；默认值问题(将一个非法值插入ENUM(也就是说，允许的值列之外的字符串)，将插入空字符串以作为特殊错误值)；索引值问题

- 尽可能不使用TEXT、BLOB类型?
- 会浪费更多的磁盘和内存空间；索引排序问题，只能使用max_sort_length的长度或者手工指定order by substring(column,length)的长度排序
- VARCHAR中会产生额外存储吗?
- VARCHAR(M)，如果M<256时会使用一个字节来存储长度，如果M>=256则使用两个字节来存储长度
- 为什么表字符集选择utf8mb4?
- 统一编码，不造成乱码；utf8mb4支持emoji表情
- 建议字段定义为NOT NULL?
- null需要更多的存储空间；使索引，索引统计变的复杂；对null处理的识货只能采用is null,is not null 而不能采用=、in、<、<>、!=、not in这些操作符号;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
- 为什么一张表中不能建立过多索引?
- innodb的二级索引使用b+tree存储，因此在update，delete，insert的时候需要调整b+tree,过多的索引会减慢更新速度
- 不建议使用%前缀模糊查询，例如LIKE “%weibo”?
- 会导致全表扫描

- 什么是覆盖索引?
- InnoDB 存储引擎中，secondary index（非主键索引）中没有直接存储行地址，存储主键值。如果用户需要查询secondary index中所不包含的数据列时，需要先通过secondary index查找到主键值，然后再通过主键查询到其他数据列，因此需要查询两次
- 覆盖索引的概念就是查询可以通过在一个索引中完成，覆盖索引效率会比较高，主键查询是天然的覆盖索引
- 合理的创建索引以及合理的使用查询语句，当使用到覆盖索引时可以获得性能提升
- 比如SELECT email,uid FROM user_email WHERE uid=xx，如果uid不是主键，适当时候可以将索引添加为index(uid,email)，以获得性能提升
- UPDATE、DELETE语句不使用LIMIT?
- 可能导致主从数据不一致；会记录到错误日志，导致日志占用大量空间
- 为什么需要避免MySQL进行隐式类型转化?
- 因为MySQL进行隐式类型转化之后，可能会将索引字段类型转化成=号右边值的类型，导致使用不到索引
- 为什么不能使用ORDER BY rand()?
- ORDER BY rand()会将数据从磁盘中读取，进行排序，会消耗大量的IO和CPU，可以在程序中获取一个rand值，然后通过在从数据库中获取对应的值

- MySQL中如何进行分页?
- 通用方式：SELECT * FROM table ORDER BY TIME DESC LIMIT 10000,10;（会导致大量的io）
- 优化方式：SELECT * FROM table inner JOIN(SELECT id FROM table ORDER BY TIME LIMIT 10000,10) as t USING(id)
- 为什么避免使用复杂的SQL?
- 将大的SQL拆分成多条简单SQL分步执行。简单的SQL容易使用到MySQL的query cache；减少锁表时间特别是MyISAM；可以使用多核cpu
- 禁止大表使用join查询，禁止大表使用子查询?
- 会产生临时表，消耗较多内存与cpu，极大影响数据库性能