建表规范

- 使用自增 `id` 作为主键
- 使用 `innodb` 引擎
- 使用统一编码 `utf8mb4`
- 尽量将所有列定义为 `NOT NULL`

- 所有表和字段都要添加注释，修改字段含义或对字段表示的状态追加时，需要及时更新字段注释
- 存储业务数据的表，建议不要物理删除，添加字段 `is_deleted` 进行逻辑删除
- 尽量避免在表中建立预留字段

- 适当进行反范式化设计，便于查询和索引优化
- 尽量做到冷热数据分离，减小表的宽度
- 单表行数超过 `500` 万行或者单表容量超过 `2GB` ，才推荐进行分库分表
- 避免在数据库中存储图片、文件等二进制数据

命名规范

- 库名与应用名保持一致
- 常用业务字段团队内部需要使用统一命名，具体需参考团队规范
- 表名和字段名，必须使用小写字母或数字，下划线分割，禁止出现数字开头，禁止两个下划线中间只出现数字见名知意，不可超过32字符
- 表名不使用复数名词
- 表达是与否概念的字段，必须使用 `is_xxx` 的方式命名，数据类型是 `UNSIGNED TINYINT` （ 1表示是，0表示否）
- 禁止使用 `MySQL` 保留字，如 `desc` 、 `range` 、 `match` 、 `delayed` 等（请参考MySQL 官方保留字）
- 主键索引使用 `pk_`前缀；唯一索引使用 `uk_`前缀；普通索引使用 `idx_`前缀
- 不同表存储相同数据的列（关联列）的列名和列类型必须完全一致
- 临时库、表名必须以 `bak_` 为前缀，并以 `_yyyyMMdd` 实时日期为后缀。例如 `tmp_test01_20190409`
- 备份库、表必须以 `bak_` 为前缀，并以 `_yyyyMMdd` 实时日期为后缀。例如 `bak_test01_20190409`

类型选择规范

- 字段可以使用多种数据类型时，优先考虑数字类型，其次为日期或二进制类型，最后是字符类型
- 整数型选择能符合需求的最短列类型，如果为非负数，必须声明 `UNSIGNED`

- 实数类型使用 `DECIMAL` ，禁止使用 `FLOAT` 和 `DOUBLE`
- 禁止使用字符串类型代替日期类型，日期占用空间小，便于查找过滤，有丰富的处理函数

- 使用 `TINYINT` 代替 `ENUM`
- 使用 `INT UNSIGNED` 存储 `IPV4`
- 使用 `VARCHAR` 时 选择能符合需求的最小长度
- 使用 `VARBINARY` 存储大小写敏感的变长字符串，`VARBINARY` 默认区分大小写，没有字符集概念，速度快

索引规范

- 业务上具有唯一特性的字段，即使是多个字段的组合，也必须建成唯一索引
- 在 `VARCHAR` 字段上建立索引时，必须指定索引长度，没必要对全字段建立索引，根据 实际文本区分度决定索引长度即可
- 每张表索引不要超过5个
- 避免建立冗余索引和重复索引
- 对于频繁查询优先考虑使用覆盖索引

SQL 编写规范

- 避免数据类型的隐式转换，会导致索引失效
- 禁止使用 `SELECT *`
- 使用 `IN` 代替 `OR`
- 不使用反向查询，如 `NOT IN` 和 `NOT LIKE`
- 禁止使用 `ORDER BY RAND()` 进行随机排序

- 禁止在 `WHERE` 从句中对列进行函数转换和计算
- 禁止 SQL 中存放业务逻辑
- 使用用 `UNION ALL` 代替 `UNION`

- 禁止在数据库中存储明文密码
- 避免使用子查询，可以把子查询优化为 `JOIN` 操作

- 超过三个表禁止 `JOIN`，需要 `JOIN` 的字段，数据类型必须绝对一致
- 超过 100w 行的批量写操作，要分批多次进行操作
- 减少与数据库的交互次数
- 合理拆分复杂的大 SQL 为多个小 SQL
- 如果有 `ORDER BY` 的场景，请注意利用索引的有序性
- 页面搜索严禁左模糊或者全模糊，如果需要请走搜索引擎来解决
- 禁止在线上数据库做压力测试
- 禁止从开发环境，测试环境直接连接生产环境数据库

数据操作规范

- 禁止使用不含有字段列表的 `INSERT` 语句
- `UPDATE` 少量数据时，先使用 `SELECT` 将需要更改的数据查出，并在 `UPDATE` 语句的 `WHERE` 条件中添加主键限制
- `UPDATE` 大量数据时，先对数据进行备份
- 禁止单条 SQL 语句同时更新多个表
- 修改字段时，将多个 `ALERT` 语句合并为一个执行
- 若需大批量插入或更新，执行语句很多，建议每隔三四百行添加 `SELECT SLEEP(1);` 语句