数据库规范

一、 库规范

1. 库名小写，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。
2. 创建数据库时必须显式指定字符集，并且字符集只能是 utf8 或者 utf8mb4。
3. 禁止使用存储过程、事件、触发器。

二、 表规范

1. 表名小写，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，一般是：**业务名称_表的作用。**
2. 表名不使用复数名词。 说明:表名应该仅仅表示表里面的实体内容，不应该表示实体数量，对应于 DO 类名也是单数 形式，符合表达习惯。
3. 临时库、表名以 tmp_ 为前缀，并以日期为后缀，备份库、表以 bak_ 为前缀，并以日期为后缀。如当日创建多个，则在日期后增加数字后缀。需要考虑定期清理。
4. 创建表时必须显式指定字符集为 utf8 或 utf8mb4。
5. 每个表及每个字段都必须提供清晰的注释。
6. 表必须有一个主键，且类型是 bigint unsigned，自增，且主键值禁止被更新。
7. 标识表里每一行主体的字段不要设为主键，建议设为其他字段如 user_id，order_id 等，并建立 unique key 索引。

8. 表必备四字段:id, is_delete,create_time, update_time。 说明:其中 id 必为主键，类型为 unsigned bigint、单表时自增、步长为 1。create_time, update_time 的类型均为 datetime 类型。is_delete 是软删除。
9. 有事务要求的必须用 InnoDB 引擎，一般也优先用 InnoDB。
10. 控制列数量，不建议超过 50 个。
11. 平衡范式与冗余：为提高效率牺牲范式设计，可适当冗余数据。
12. 表的索引数建议不超过 8 个。
13. 禁止使用外键。

三、 字段规范

1. 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。
2. 字段名尽可能复用前面的定好的字段名，类型尽可能相同。
3. 字段名尽量避免用关键字。
4. 如果修改字段含义或对字段表示的状态追加时，必须及时更新字段注释。
5. 表与表之间的相关联字段名称要求尽可能的相同。
6. 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)。
7. 如使用缩写，请尽量名字易懂简短，如 description --> desc;information --> info;address --> addr 等。

8. 字段尽量设置为 NOT NULL， 为字段提供默认值。 如字符型的默认值为一个空字符值串’’;数值型默认值为数值 0;逻辑型的默认值为数值 0;

四、 字段类型规范

1. 用好数值类型，能使用 tinyint 就不要使用 smallint,int，能用数值类型就不要用字符串,非负的数字类型字段,都添加上 UNSINGED。

类型	字节	表示范围
tinyint	1	无符号值:0～255;有符号值： -128~127
smallint	2	无符号值:0～65536;有符号值： -32768~32767
mediumint	3	无符号值:0～16777215;有符号值： -8388608~8388607
int	4	无符号值:0~4294967295;有符号值： -2147483648~2147483647
bigint	8	无符号值:0~((2³²×²)-1);有符号值： -(2³²×²)/2 ~ (2³²×²)/2-1

2. 禁止使用 ENUM，可使用 TINYINT 代替。

3. 能用 char 就不要用 varchar，能用 varchar 就不要用 text。

4. 少用 text/blob。

   varchar 的性能会比 text 高很多
   实在避免不了 blob，请拆表

5. 使用 varchar(20)存储手机号。

   涉及到区号或者国家代号，可能出现+-()
   varchar 可以支持模糊查询

6. 小数类型要为 decimal，禁止使用 float 和 double。
7. 尽量不用字符串存储日期型的数据，优先使用 datetime。存储年使用 year 类型，存储日期使用 date 类型。
8. IPV4 要用 int unsigned 存放。
9. 不在数据库里存图片。

五、 索引规范

1. 命名简洁明确，主键索引名为 pk_字段名；唯一索引名为 uk_字段名；普通索引名则为 idx_字段名。

2. 联合索引的字段数控制在 5 个以内。

3. 联合索引的字段排列顺序以去重后字段的数值的个数大小排序先后顺序。比如表 mk_task 有 id,name，id 有 50000 个独立值，name 有 5000 个独立值，那么，顺序是 id 在 name 前面，建立的索引是 idx_id_name。

4. 合理创建联合索引（避免冗余），(a,b,c) 相当于（a）、（a、b）、（a、b、c）。

5. 只给常用的查询条件加索引。

6. order by、distinct、group by 后的字段尽量建立索引。

7. 过滤性高的列建索引，取值范围固定的列不建索引。

8. 唯一的记录添加唯一索引。

9. 频繁更新的列不要建索引。

10. 不要对索引列运算。

11. 合理利用组合索引，注意索引字段先后顺序。

12. 多列组合索引，过滤性高的字段最前。

13. order by 字段建立索引，避免 filesort。

14. 防止因字段类型不同造成的隐式转换，导致索引失效。

15. 在 varchar 字段上建立索引时，必须指定索引长度。

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。 说明:索引的长度与区分度是一对矛盾体，一般对字符串类型数据，长度为 20 的索引，区分度会高达 90%以上，可以使用 count(distinct left(列名，索引长度))/count(*)的区分度来确定。

六、 SQL 规范

1. 禁止使用 SELECT *，只获取必要的字段。

2. 不要使用 count(列名)或 count(常量)来替代 count()，count()是 SQL92 定义的 标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。

count(*)会统计值为 NULL 的行，而 count(列名)不会统计此列为 NULL 值的行。

3. count(distinct col) 计算该列除 NULL 之外的不重复行数

   count(distinct col1, col2) 如果其中一列全为 NULL，那么即使另一列有不同的值，也返回为 0。

4. 禁止使用属性隐式转换，会导致索引失效。

   SELECT uid FROM user WHERE phone=13812345678 会导致全表扫描，而不能命中 phone 索引。

5. 禁止在 WHERE 条件的属性上使用函数或者表达式，会导致索引失效。

6. update、delete 的 where 尽量使用有索引的字段或主键。

7. 事务里更新语句尽量基于主键或 unique key，如 update … where id=XX。

8. 禁止负向查询，以及%开头的模糊查询。

   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

9. 禁止跨 db 的 join 语句。

10. 禁止大表使用 join 查询，禁止大表使用子查询。

11. 不建议使用子查询，建议将子查询 SQL 拆开结合程序多次查询，或使用 join 来代替子查询。

12. 在多表 join 中，尽量选取结果集较小的表作为驱动表，来 join 其他表。

13. 尽量减少使用 order by，和业务沟通能不排序就不排序，或将排序放到程序端去做。order by、group by、distinct 这些语句较为耗费 CPU。

14. order by、group by、distinct 这些 SQL 尽量利用索引直接检索出排序好的数据。如 where a=1 order by b 可以利用 key(a,b)

15. OR 改写为 IN。

```
or 的效率是 n 级别
in 的消息时 log(n)级别
in 的个数建议控制在 1000 以内
select id from user where phone='15900000001' or phone='13600000001';
=>
select id from user where phone in ('15900000001', '13600000001');
```

16. OR 改写为 UNION。

```
mysql 的索引合并很弱智
select id from t where phone = '15900000001' or name = 'john';
=>
select id from t where phone='15900000001'
union
```

select id from t where name='jonh'

17. 禁止使用 INSERT INTO xxx VALUES(yyy)，必须显示指定插入的列属性。

18. 当只要一行数据时使用 LIMIT 1。

19. 对于连续的数值，使用 BETWEEN 代替 IN

20. 分页优化

利用延迟关联或者子查询优化超多分页场景。
说明:MySQL 并不是跳过 offset 行，而是取 offset+N 行，然后返回放弃前 offset 行，返回 N 行，那当 offset 特别大的时候，效率就非常的低下，要么控制返回的总页数，要么对超过特定阈值的页数进行 SQL 改写。
正例:先快速定位需要获取的 id 段，然后再关联:
SELECT a.id,a.info FROM 表 A, (select id from 表 A where 条件 LIMIT 100000,20) b where a.id=b.id

21. 事务尽量简单，同一事务里的 sql 语句尽量不超 5 个。

22. 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

23. SQL 语句尽可能简单。

24. 尽量不在数据库做运算，尽量移至业务层。

25. 拒绝 3B(拒绝大 SQL 语句，拒绝大事物，拒绝大批量)。

26. 禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。

27. 超过三个表禁止 join。需要 join 的字段，数据类型必须绝对一致；多表关联查询时，保证被关联的字段需要有索引。

28. 当某一列 col 的值全是 NULL 时，count(col)的返回结果为 0，但 sum(col)的返回结果为 NULL，因此使用 sum()时需注意 NPE 问题。

    可以使用如下方式来避免 sum 的 NPE 问题:SELECT IF(ISNULL(SUM(g)),0,SUM(g)) FROM table;

29. 使用 ISNULL()来判断是否为 NULL 值。

    - NULL 与任何值的直接比较都为 NULL。
    - NULL<>NULL 的返回结果是 NULL，而不是 false。
    - NULL=NULL 的返回结果是 NULL，而不是 true。
    - NULL<>1 的返回结果是 NULL，而不是 true。

30. 可以多考虑使用预编译语句进行数据库操作。