# 数据库开发规范

## 基础规范

1. 使用INNODB存储引擎
2. 表字符集使用UTF8
3. 所有表都需要添加注释
4. 单表数据量建议控制在5000W以内
5. 不在数据库中存储图像、文件等大数据
6. 禁止在线上做数据库压力测试
7. 禁止从测试、开发环境直连数据库

## 命名规则

1. 库名、表名、字段名必须使用小写字母,并采用下划线分割。
2. 所有数据库对象禁止超过32个字符。
3. 所有数据库对象必须见名知意。命名与业务、产品线等相关联。
4. 所有数据库对象禁止使用MySQL保留字（参考附件《MYSQL保留字》）。
5. 临时库、表名必须以tmp为前缀，并以日期为后缀。例如 tmp\_mydb\_20130704。
6. 备份库、表必须以bak为前缀，并以日期为后缀。例如 bak\_mydb\_20130704。
7. 所有数据库对象加上相应前缀，如下表：

表格 1数据库对象相应前缀

|  |  |
| --- | --- |
| 数据库对象 | 前缀 |
| 数据库(Database) | db\_ |
| 表 (Table) | tbl\_ |
| 字段(Column) | 无 |
| 视图 (View) | viw\_ |
| 存储过程 (Stored procedure) | prd\_ |
| 触发器(Trigger) | trg\_ |
| 索引(Index) | idx\_ |
| 主键(Primary key) | pk\_ |
| 外键(Foreign key) | fk\_ |
| Check 约束(Check Constraint) | ck\_ |
| Default 约束(Default Constraint) | df\_ |
| 用户定义数据类型 (User-defined data type) | udt\_ |
| 用户定义函数 (User-defined function) | fun\_ |

## 库、表、字段开发设计规范

1. 禁止使用分区表
2. 拆分大字段和访问频率低的字段，分离冷热数据
3. 用HASH进行散表，表名后缀使用十进制数，下标从0开始
4. 按日期时间分表需符合YYYY[MM][DD][HH]格式,例如2013071601。年份必须用4位数字表示。例如按日散表user\_20110209、 按月散表user\_201102。
5. 采用合适的分库分表策略。例如千库十表、十库百表等
6. 尽可能不使用TEXT、BLOB类型
7. 用DECIMAL代替FLOAT和DOUBLE存储精确浮点数。例如与货币、金融相关的数据。
8. 越简单越好：将字符转化为数字、使用TINYINT来代替ENUM类型
9. 所有字段均定义为NOT NULL
10. 使用UNSIGNED存储非负整数
11. INT类型固定占用4字节存储
12. 使用timestamp存储时间
13. 使用INT UNSIGNED存储IPV4
14. 使用VARBINARY存储大小写敏感的变长字符串或二进制内容
15. 禁止在数据库中存储明文密码，把密码加密后存储
16. 使用尽可能小的VARCHAR字段。VARCHAR(N)中的N表示字符数而非字节数。
17. 区分使用DATETIME和TIMESTAMP。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TIMESTAMP类型。
18. 用好数值类型字段

Tinyint (1Byte)

smallint (2Byte)

mediumint (3Byte)

int (4Byte)

bigint (8Byte)

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 字节 | 最小值 | 最大值 |
|  |  | (带符号的/无符号的) | (带符号的/无符号的) |
| TINYINT | 1 | -128 | 127 |
| 无符号 |  | 0 | 255 |
| SMALLINT | 2 | -32768 | 32767 |
| 无符号 |  | 0 | 65535 |
| MEDIUMINT | 3 | -8388608 | 8388607 |
| 无符号 |  | 0 | 16777215 |
| INT | 4 | -2147483648 | 2147483647 |
| 无符号 |  | 0 | 4294967295 |
| BIGINT | 8 | -9223372036854775808 | 9223372036854775807 |
| 无符号 |  | 0 | 18446744073709551615 |

如果数值字段没有那么大，就不要用 bigint

1. 存储ip最好用int存储而非char(15)
2. 不允许使用ENUM
3. 避免使用NULL字段，NULL字段很难查询优化，NULL字段的索引需要额外空间，NULL字段的复合索引无效
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. 数据库中不允许存储大文件，或者照片，可以将大对象放到磁盘上，数据库中存储它的路径

## 索引规范

1、索引的数量要控制：

1. 单张表中索引数量不超过5个
2. 单个索引中的字段数不超过5个
3. 对字符串使用前缀索引，前缀索引长度不超过8个字符
4. 建议优先考虑前缀索引，必要时可添加伪列并建立索引
5. 索引名必须全部使用小写。
6. 非唯一索引按照“idx\_字段名称[\_字段名称]”进用行命名。例如idx\_age\_name。
7. 唯一索引按照“uniq\_字段名称[\_字段名称]”进用行命名。例如uniq\_age\_name。
8. 组合索引建议包含所有字段名,过长的字段名可以采用缩写形式。例如idx\_age\_name\_add。
9. 表必须有主键,推荐使用UNSIGNED自增列作为主键。
10. 唯一键由3个以下字段组成,并且字段都是整形时,可使用唯一键作为主键。其他情况下,建议使用自增列或发号器作主键。
11. 禁止使用外键。
12. 联表查询时,JOIN列的数据类型必须相同,并且要建立索引。
13. 不在低基数列上建立索引,例如“性别”。
14. 选择区分度大的列建立索引。组合索引中,区分度大的字段放在最前。
15. 对字符串使用前缀索引,前缀索引长度不超过8个字符。
16. 不对过长的VARCHAR字段建立索引。建议优先考虑前缀索引,或添加CRC32或MD5伪列并建立索引。
17. 合理创建联合索引,(a,b,c) 相当于 (a) 、(a,b) 、(a,b,c)。
18. 合理使用覆盖索引减少IO,避免排序。

2、主键准则

(1) 表必须有主键

(2) 不使用更新频繁的列作为主键

(3) 尽量不选择字符串列作为主键

(4) 不使用UUID MD5 HASH这些作为主键(数值太离散了)

(5) 默认使非空的唯一键作为主键

(6) 建议选择自增或发号器

3、重要的SQL必须被索引，比如：

(1) UPDATE、DELETE语句的WHERE条件列

(2) ORDER BY、GROUP BY、DISTINCT的字段

4、多表JOIN的字段注意以下：

(1) 区分度最大的字段放在前面

(2) 核心SQL优先考虑覆盖索引

(3) 避免冗余和重复索引

(4) 索引要综合评估数据密度和分布以及考虑查询和更新比例

5、索引禁忌

(1) 不在低基数列上建立索引，例如“性别”

(2) 不在索引列进行数学运算和函数运算

6、尽量不使用外键

(1) 外键用来保护参照完整性，可在业务端实现

(2) 对父表和子表的操作会相互影响，降低可用性

7、索引命名：非唯一索引必须以 idx\_字段1\_字段2命名，唯一所以必须以uniq\_字段1\_字段2命名，索引名称必须全部小写

8、新建的唯一索引必须不能和主键重复

9、索引字段的默认值不能为NULL，要改为其他的default或者空。NULL非常影响索引的查询效率

10、反复查看与表相关的SQL，符合最左前缀的特点建立索引。多条字段重复的语句，要修改语句条件字段的顺序，为其建立一条联合索引，减少索引数量

11、能使用唯一索引就要使用唯一索引，提高查询效率

12、研发要经常使用explain，如果发现索引选择性差，必须让他们学会使用hint

## SQL规范

(1) sql语句尽可能简单，大的sql想办法拆成小的sql语句(充分利用QUERY CACHE和充分利用多核CPU)

(2) 事务要简单，整个事务的时间长度不要太长

(3) 避免使用触发器、函数、存储过程

(4) 降低业务耦合度，为sacle out、sharding留有余地

(5) 避免在数据库中进⾏数学运算(MySQL不擅长数学运算和逻辑判断)

(4) 不要用select \*，查询哪几个字段就select 这几个字段

(5) sql中使用到OR的改写为用 IN() (or的效率没有in的效率高)

(6) in里面数字的个数建议控制在1000以内

(7) limit分页注意效率。Limit越大，效率越低。可以改写limit，比如例子改写：

select id from tlimit 10000, 10; => select id from t where id > 10000 limit10;

(9) 使用union all替代union

(10) 避免使⽤大表的JOIN

(11) 使用group by 分组、自动排序

(12) 对数据的更新要打散后批量更新，不要一次更新太多数据

(13) 减少与数据库的交互次数

(13) 注意使用性能分析工具

Sql explain / showprofile / mysqlsla

(14) SQL语句要求所有研发，SQL关键字全部是大写，每个词只允许有一个空格

(15) SQL语句不可以出现隐式转换，比如 select id from 表 where id='1'

(16) IN条件里面的数据数量要少，我记得应该是500个以内，要学会使用exist代替in，exist在一些场景查询会比in快

(17) 能不用NOT IN就不用NOTIN，坑太多了。。会把空和NULL给查出来

(18) 在SQL语句中，禁止使用前缀是%的like

(19) 不使用负向查询，如not in/like

(19) 关于分页查询：程序里建议合理使用分页来提高效率limit，offset较大要配合子查询使用

(20) 禁止在数据库中跑大查询

(21) 使用预编译语句，只传参数，比传递SQL语句更高效；一次解析，多次使用；降低SQL注入概率

(22) 禁止使用order by rand()

(23) 禁止单条SQL语句同时更新多个表

## 流程规范

(1) 所有的建表操作需要提前告知该表涉及的查询sql；

(2) 所有的建表需要确定建立哪些索引后才可以建表上线；

(3) 所有的改表结构、加索引操作都需要将涉及到所改表的查询sql发出来告知DBA等相关人员；

(4) 在建新表加字段之前，要求研发至少要提前3天邮件出来，给dba们评估、优化和审核的时间

(5)批量导入、导出数据必须提前通知DBA协助观察

(6) 禁止在线上从库执行后台管理和统计类查询

(7) 禁止有super权限的应用程序账号存在

(8) 推广活动或上线新功能必须提前通知DBA进行行流量评估

(9) 不在业务高峰期批量更新、查询数据库

## FAQ

**1.库名、表名、字段名必须使用小写字母,并采用下划线分割。**

a)MySQL有配置参数lower\_case\_table\_names,不可动态更改,linux系统默认为 0,即库表名以实际情况存储,大小写敏感。如果是1,以小写存储,大小写不敏感。如果是2,以实际情况存储,但以小写比较。

b)如果大小写混合使用,可能存在abc,Abc,ABC等多个表共存,容易导致混乱。

c)字段名显式区分大小写,但实际使用不区分,即不可以建立两个名字一样但大小写不一样的字段。

d)为了统一规范, 库名、表名、字段名使用小写字母。

**2.库名、表名、字段名禁止超过32个字符。**

库名、表名、字段名支持最多64个字符,但为了统一规范、易于辨识以及减少传输量,禁止超过32个字符。

**3.使用INNODB存储引擎。**

INNODB引擎是MySQL5.5版本以后的默认引擘,支持事务、行级锁,有更好的数据恢复能力、更好的并发性能,同时对多核、大内存、SSD等硬件支持更好,支持数据热备份等,因此INNODB相比MyISAM有明显优势。

**4.库名、表名、字段名禁止使用MySQL保留字。**

当库名、表名、字段名等属性含有保留字时,SQL语句必须用反引号引用属性名称,这将使得SQL语句书写、SHELL脚本中变量的转义等变得非常复杂。

**5.禁止使用分区表。**

分区表对分区键有严格要求;分区表在表变大后,执行DDL、SHARDING、单表恢复等都变得更加困难。因此禁止使用分区表,并建议业务端手动SHARDING。

**6.建议使用UNSIGNED存储非负数值。**

同样的字节数,非负存储的数值范围更大。如TINYINT有符号为 -128-127,无符号为0-255。

**7.建议使用INT UNSIGNED存储IPV4。**

UNSINGED INT存储IP地址占用4字节,CHAR(15)则占用15字节。另外,计算机处理整数类型比字符串类型快。使用INT UNSIGNED而不是CHAR(15)来存储IPV4地址,通过MySQL函数inet\_ntoa和inet\_aton来进行转化。IPv6地址目前没有转化函数,需要使用DECIMAL或两个BIGINT来存储。

例如:

SELECT INET\_ATON('209.207.224.40'); 3520061480

SELECT INET\_NTOA(3520061480); 209.207.224.40

**8.强烈建议使用TINYINT来代替ENUM类型。**

ENUM类型在需要修改或增加枚举值时,需要在线DDL,成本较大;ENUM列值如果含有数字类型,可能会引起默认值混淆。

**9.使用VARBINARY存储大小写敏感的变长字符串或二进制内容。**

VARBINARY默认区分大小写,没有字符集概念,速度快。

**10.INT类型固定占用4字节存储,例如INT(4)仅代表显示字符宽度为4位,不代表存储长度。**

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,python、java客户端等不具备这个功能。

**11.区分使用DATETIME和TIMESTAMP。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TIMESTAMP类型。

DATETIME和TIMESTAMP都是精确到秒,优先选择TIMESTAMP,因为TIMESTAMP只有4个字节,而DATETIME8个字节。同时TIMESTAMP具有自动赋值以及自动更新的特性。注意:在5.5和之前的版本中,如果一个表中有多个timestamp列,那么最多只能有一列能具有自动更新功能。

如何使用TIMESTAMP的自动赋值属性?

a)自动初始化,并自动更新: column1 TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

b)只是自动初始化: column1 TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

c)自动更新,初始化的值为0: column1 TIMESTAMP DEFAULT 0 ON UPDATE CURRENT\_TIMESTAMP

d)初始化的值为0: column1 TIMESTAMP DEFAULT 0

**12.所有字段均定义为NOT NULL。**

a)对表的每一行,每个为NULL的列都需要额外的空间来标识。

b)B树索引时不会存储NULL值,所以如果索引字段可以为NULL,索引效率会下降。

c)建议用0、特殊值或空串代替NULL值。

**13.将大字段、访问频率低的字段拆分到单独的表中存储,分离冷热数据。**

有利于有效利用缓存,防止读入无用的冷数据,较少磁盘IO,同时保证热数据常驻内存提高缓存命中率。

**14.禁止在数据库中存储明文密码。**

采用加密字符串存储密码，并保证密码不可解密，同时采用随机字符串加盐保证密码安全。防止数据库数据被公司内部人员或黑客获取后，采用字典攻击等方式暴力破解用户密码。

**15.表必须有主键,推荐使用UNSIGNED自增列作为主键。**

表没有主键,INNODB会默认设置隐藏的主键列;没有主键的表在定位数据行的时候非常困难,也会降低基于行复制的效率。

**16.禁止冗余索引。**

索引是双刃剑,会增加维护负担,增大IO压力。(a,b,c)、(a,b),后者为冗余索引。可以利用前缀索引来达到加速目的,减轻维护负担。

**17.禁止重复索引。**

primary key a;uniq index a;重复索引增加维护负担、占用磁盘空间,同时没有任何益处。

**18.不在低基数列上建立索引,例如“性别”。**

大部分场景下,低基数列上建立索引的精确查找,相对于不建立索引的全表扫描没有任何优势,而且增大了IO负担。

**19.合理使用覆盖索引减少IO,避免排序。**

覆盖索引能从索引中获取需要的所有字段,从而避免回表进行二次查找,节省IO。INNODB存储引擎中, secondary index(非主键索引,又称为辅助索引、二级索引)没有直接存储行地址,而是存储主键值。如果用户需要查询secondary index中所不包含的数据列,则需要先通过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。覆盖索引则可以在一个索引中获取所有需要的数据,因此效率较高。主键查询是天然的覆盖索引。例如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。

**20.用IN代替OR。SQL语句中IN包含的值不应过多,应少于1000个。**

IN是范围查找,MySQL内部会对IN的列表值进行排序后查找,比OR效率更高。

**21.表字符集使用UTF8,必要时可申请使用UTF8MB4字符集。**

a)UTF8字符集存储汉字占用3个字节,存储英文字符占用一个字节。

b)UTF8统一而且通用,不会出现转码出现乱码风险。

c)如果遇到EMOJ等表情符号的存储需求,可申请使用UTF8MB4字符集。

**22.用UNION ALL代替UNION。**

UNION ALL不需要对结果集再进行排序。

**23.禁止使用order by rand()。**

order by rand()会为表增加一个伪列,然后用rand()函数为每一行数据计算出rand()值,然后基于该行排序, 这通常都会生成磁盘上的临时表,因此效率非常低。建议先使用rand()函数获得随机的主键值,然后通过主键获取数据。

**24.建议使用合理的分页方式以提高分页效率。**

第一种分页写法：

select \* from t

where thread\_id = 771025

and deleted = 0

order by gmt\_create asc limit 0, 15;

原理：一次性根据过滤条件取出所有字段进行排序返回。

数据访问开销=索引IO+索引全部记录结果对应的表数据IO

缺点：该种写法越翻到后面执行效率越差，时间越长，尤其表数据量很大的时候。

适用场景：当中间结果集很小（10000行以下）或者查询条件复杂（指涉及多个不同查询字段或者多表连接）时适用。

第二种分页写法：

select t.\* from (

select id from t

where thread\_id = 771025 and deleted = 0 order by gmt\_create asc limit 0, 15) a, t

where a.id = t.id;

前提：假设t表主键是id列，且有覆盖索引secondary key:(thread\_id, deleted, gmt\_create)

原理：先根据过滤条件利用覆盖索引取出主键id进行排序，再进行join操作取出其他字段。

数据访问开销=索引IO+索引分页后结果（例子中是15行）对应的表数据IO。

优点：每次翻页消耗的资源和时间都基本相同，就像翻第一页一样。

适用场景：当查询和排序字段（即where子句和order by子句涉及的字段）有对应覆盖索引时，且中间结果集很大的情况时适用。

**25.SELECT只获取必要的字段,禁止使用SELECT \*。**

减少网络带宽消耗;

能有效利用覆盖索引;

表结构变更对程序基本无影响。

**26.SQL中避免出现now()、rand()、sysdate()、current\_user()等不确定结果的函数。**

语句级复制场景下,引起主从数据不一致;不确定值的函数,产生的SQL语句无法利用QUERY CACHE。

**27.采用合适的分库分表策略。例如千库十表、十库百表等。**

采用合适的分库分表策略,有利于业务发展后期快速对数据库进行水平拆分,同时分库可以有效利用MySQL的多线程复制特性。

**28.减少与数据库交互次数,尽量采用批量SQL语句。**

使用下面的语句来减少和db的交互次数:

a)INSERT ... ON DUPLICATE KEY UPDATE

b)REPLACE INTO

c)INSERT IGNORE

d)INSERT INTO VALUES()

**29.拆分复杂SQL为多个小SQL,避免大事务。**

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。

**30.对同一个表的多次alter操作必须合并为一次操作。**

mysql对表的修改绝大部分操作都需要锁表并重建表,而锁表则会对线上业务造成影响。为减少这种影响,必须把对表的多次alter操作合并为一次操作。例如,要给表t增加一个字段b,同时给已有的字段aa建立索引, 通常的做法分为两步:

alter table t add column b varchar(10);

然后增加索引:

alter table t add index idx\_aa(aa);

正确的做法是:

alter table t add column b varchar(10),add index idx\_aa(aa);

**31.避免使用存储过程、触发器、视图、自定义函数等。**

这些高级特性有性能问题,以及未知BUG较多。业务逻辑放到数据库会造成数据库的DDL、SCALE OUT、 SHARDING等变得更加困难。

**32.禁止有super权限的应用程序账号存在。**

安全第一。super权限会导致read only失效,导致较多诡异问题而且很难追踪。

**33.提交线上建表改表需求,必须详细注明涉及到的所有SQL语句(包括INSERT、DELETE、**UPDATE),便于DBA进行审核和优化。

并不只是SELECT语句需要用到索引。UPDATE、DELETE都需要先定位到数据才能执行变更。因此需要业务提供所有的SQL语句便于DBA审核。

**34.不要在MySQL数据库中存放业务逻辑。**

数据库是有状态的服务,变更复杂而且速度慢,如果把业务逻辑放到数据库中,将会限制业务的快速发展。建议把业务逻辑提前,放到前端或中间逻辑层,而把数据库作为存储层,实现逻辑与存储的分离。

附件：《MySQL保留字》

## MySQL保留字

|  |  |  |
| --- | --- | --- |
| **保留字** | **保留字** | **保留字** |
| ADD | ALL | ALTER |
| ANALYZE | AND | AS |
| ASC | AUTO\_INCREMENT | BDB |
| BEFORE | BERKELEYDB | BETWEEN |
| BIGINT | BINARY | BLOB |
| BOTH | BTREE | BY |
| CASCADE | CASE | CHANGE |
| CHAR | CHARACTER | CHECK |
| COLLATE | COLUMN | COLUMNS |
| CONSTRAINT | CREATE | CROSS |
| CURRENT\_DATE | CURRENT\_TIME | CURRENT\_TIMESTAMP |
| DATABASE | DATABASES | DAY\_HOUR |
| DAY\_MINUTE | DAY\_SECOND | DEC |
| DECIMAL | DEFAULT | DELAYED |
| DELETE | DESC | DESCRIBE |
| DISTINCT | DISTINCTROW | DIV |
| DOUBLE | DROP | ELSE |
| ENCLOSED | ERRORS | ESCAPED |
| EXISTS | EXPLAIN | FALSE |
| FIELDS | FLOAT | FOR |
| FORCE | FOREIGN | FROM |
| FULLTEXT | FUNCTION | GRANT |
| GROUP | HASH | HAVING |
| HIGH\_PRIORITY | HOUR\_MINUTE | HOUR\_SECOND |
| IF | IGNORE | IN |
| INDEX | INFILE | INNER |
| INNODB | INSERT | INT |
| INTEGER | INTERVAL | INTO |
| IS | JOIN | KEY |
| KEYS | KILL | LEADING |
| LEFT | LIKE | LIMIT |
| LINES | LOAD | LOCALTIME |
| LOCALTIMESTAMP | LOCK | LONG |
| LONGBLOB | LONGTEXT | LOW\_PRIORITY |
| MASTER\_SERVER\_ID | MATCH | MEDIUMBLOB |
| MEDIUMINT | MEDIUMTEXT | MIDDLEINT |
| MINUTE\_SECOND | MOD | MRG\_MYISAM |
| NATURAL | NOT | NULL |
| NUMERIC | ON | OPTIMIZE |
| OPTION | OPTIONALLY | OR |
| ORDER | OUTER | OUTFILE |
| PRECISION | PRIMARY | PRIVILEGES |
| PROCEDURE | PURGE | READ |
| REAL | REFERENCES | REGEXP |
| RENAME | REPLACE | REQUIRE |
| RESTRICT | RETURNS | REVOKE |
| RIGHT | RLIKE | RTREE |
| SELECT | SET | SHOW |
| SMALLINT | SOME | SONAME |
| SPATIAL | SQL\_BIG\_RESULT | SQL\_CALC\_FOUND\_ROWS |
| SQL\_SMALL\_RESULT | SSL | STARTING |
| STRAIGHT\_JOIN | STRIPED | TABLE |
| TABLES | TERMINATED | THEN |
| TINYBLOB | TINYINT | TINYTEXT |
| TO | TRAILING | TRUE |
| TYPES | UNION | UNIQUE |
| UNLOCK | UNSIGNED | UPDATE |
| USAGE | USE | USER\_RESOURCES |
| USING | VALUES | VARBINARY |
| VARCHAR | VARCHARACTER | VARYING |
| WARNINGS | WHEN | WHERE |
| WITH | WRITE | XOR |
| YEAR\_MONTH | ZEROFILL |  |

附件：《项目建库建议》

# 项目建库建议

1. 数据库名使用“qs\_”前缀+项目系统名称，如“qs\_oa”，以避免与其他非本公司创建的数据库发生冲突；
2. 数据库表名使用 “qs\_”+业务名称，如“qs\_user”，以避免与外接系统业务表发生冲突；
3. 所有对象都使用小写字母，所有对象名称使用(“前缀\_”+)“业务系统相关的英文单词名称命名”，多个单词之间使用下划线分割；
4. 信息增加型表必含字段：

id INT：主键使用MySQL的自增类型

add\_time DATETIME：创建日期（大字段拆分表除外），必须使用数据库时间（用now()生成）

update\_time TIMESTAMP：修改日期（大字段拆分表除外, 但内容变化必须修改主表的 update\_time字段），由数据库自动变更（CURRENT\_TIMESTAMP），应用不操作此字段

以上3个字段都必须是没有任何商业意义的与业务无关的字段，不允许赋予任何商业意义