

# 去哪儿MySQL开发规范

版本	修改内容	时间	负责人
1.0	初稿	2013-07-18	DBA
1.5	讨论稿，组内讨论修正	2013-07-23	DBA
2.0	正式版	2013-07-30	DBA

## 1.命名规范

- (1)库名、表名、字段名必须使用小写字母，并采用下划线分割。【FAQ】
- (2)库名、表名、字段名禁止超过32个字符。【FAQ】
- (3)库名、表名、字段名必须见名知意。命名与业务、产品线等相关联。【产品线对应名称说明(供参考)】
- (4)库名、表名、字段名禁止使用MySQL保留字。【FAQ】【MySQL保留字列表】
- (5)临时库、表名必须以tmp为前缀，并以日期为后缀。例如 tmp\_test01\_20130704。
- (6)备份库、表必须以bak为前缀，并以日期为后缀。例如 bak\_test01\_20130704。

## 2.基础规范

- (1)使用INNODB存储引擎。【FAQ】
- (2)表字符集使用UTF8，必要时可申请使用UTF8MB4字符集。【FAQ】
- (3)所有表都需要添加注释；除主键外的其他字段都需要增加注释。推荐采用英文标点，避免出现乱码。【建表语句示例】
- (4)禁止在数据库中存储图片、文件等大数据。
- (5)每张表数据量建议控制在5000W以内。
- (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。

## 3.库表设计

- (1)禁止使用分区表。【FAQ】
- (2)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。【FAQ】
- (3)推荐使用HASH进行散表，表名后缀使用十进制数，数字必须从0开始。
- (4)按日期时分表需符合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。【FAQ】

## 4.字段设计

- (1)建议使用UNSIGNED存储非负数值。【FAQ】
- (2)建议使用INT UNSIGNED存储IPV4。【FAQ】
- (3)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。
- (4)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。【FAQ】

- (5)区分使用TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT数据类型。例如取值范围为0-80时，使用TINYINT UNSIGNED。 [【数据类型存储空间需求说明】](#)
- (6)强烈建议使用TINYINT来代替ENUM类型。 [【FAQ】](#)
- (7)尽可能不使用TEXT、BLOB类型。
- (8)禁止在数据库中存储明文密码。 [【FAQ】](#)
- (9)使用VARBINARY存储大小写敏感的变长字符串或二进制内容。 [【FAQ】](#)
- (10)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。
- (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。 [【FAQ】](#)
- (12)所有字段均定义为NOT NULL。 [【FAQ】](#)

## 5.索引规范

- (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)索引名必须全部使用小写。
- (4)非唯一索引按照“idx\_字段名称[\_字段名称]”进行命名。例如idx\_age\_name。
- (5)唯一索引按照“uniq\_字段名称[\_字段名称]”进行命名。例如uniq\_age\_name。
- (6)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。
- (7)表必须有主键，推荐使用UNSIGNED自增列作为主键。 [【FAQ】](#)
- (8)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。
- (9)禁止冗余索引。 [【FAQ】](#)
- (10)禁止重复索引。 [【FAQ】](#)
- (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”。 [【FAQ】](#)
- (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。 [【FAQ】](#)

## 6.SQL设计

- (1)使用prepared statement，可以提升性能并避免SQL注入。
- (2)用IN代替OR。SQL语句中IN包含的值不应过多，应少于1000个。 [【FAQ】](#)
- (3)禁止隐式转换。数值类型禁止加引号；字符串类型必须加引号。
- (4)避免使用JOIN和子查询。必要时推荐用JOIN代替子查询。
- (5)避免在MySQL中进行数学运算和函数运算。
- (6)减少与数据库交互次数，尽量采用批量SQL语句。 [【FAQ】](#)
- (7)拆分复杂SQL为多个小SQL，避免大事务。 [【FAQ】](#)
- (8)获取大量数据时，建议分批次获取数据，每次获取数据少于2000条，结果集应小于1M。
- (9)用UNION ALL代替UNION。 [【FAQ】](#)
- (10)统计行数用COUNT(\*)。
- (11)SELECT只获取必要的字段，禁止使用SELECT \*。 [【FAQ】](#)
- (12)SQL中避免出现now()、rand()、sysdate()、current\_user()等不确定结果的函数。 [【FAQ】](#)
- (13)INSERT语句必须指定字段列表，禁止使用INSERT INTO TABLE()。
- (14)禁止单条SQL语句同时更新多个表。

- (15)避免使用存储过程、触发器、视图、自定义函数等。 [【FAQ】](#)
- (16)建议使用合理的分页方式以提高分页效率。 [【FAQ】](#)
- (17)禁止在从库上执行后台管理和统计类功能的QUERY，必要时申请统计类从库。
- (18)程序应有捕获SQL异常的处理机制，必要时通过rollback显式回滚。
- (19)重要SQL必须被索引：update、delete的where条件列、order by、group by、distinct字段、多表join字段。
- (20)禁止使用%前导查询，例如：like “%abc”，无法利用到索引。
- (21)禁止使用负向查询，例如 not in、!=、not like。
- (22)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。
- (23)禁止使用order by rand()。 [【FAQ】](#)

## 7.行为规范

- (1)表结构变更必须通知DBA进行审核。
- (2)禁止有super权限的应用程序账号存在。 [【FAQ】](#)
- (3)禁止有DDL、DCL权限的应用程序账号存在。
- (4)重大项目的数据库方案选型和设计必须提前通知DBA参与。
- (5)批量导入、导出数据必须通过DBA审核，并在执行过程中观察服务。
- (6)批量更新数据，如UPDATE、DELETE操作，必须DBA进行审核，并在执行过程中观察服务。
- (7)产品出现非数据库导致的故障时，如被攻击，必须及时通DBA，便于维护服务稳定。
- (8)业务部门程序出现BUG等影响数据库服务的问题，必须及时通知DBA，便于维护服务稳定。
- (9)业务部门推广活动或上线新功能，必须提前通知DBA进行服务和访问量评估，并留出必要时间以便DBA完成扩容。
- (10)出现业务部门人为误操作导致数据丢失，需要恢复数据的，必须第一时间通知DBA，并提供准确时间点、误操作语句等重要线索。
- (11)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。 [【FAQ】](#)
- (12)对同一个表的多次alter操作必须合并为一次操作。 [【FAQ】](#)
- (13)不要在MySQL数据库中存放业务逻辑。 [【FAQ】](#)

## 8.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)字段名显示区分大小写，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。 [【返回】](#)

### 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。

[【返回】](#)

### 3.使用INNODB存储引擎。

INNODB引擎是MySQL5.5版本以后的默认引擎，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。 [【返回】](#)

#### 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。[【返回】](#)

#### 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。[【返回】](#)

#### 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。[【返回】](#)

#### 7.建议使用INT UNSIGNED存储IPV4。

用UNSIGNED 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 【返回】
```

#### 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，成本较高；ENUM列值如果含有数字类型，可能会引起默认值混淆。[【ENUM类型说明】](#) [【返回】](#)

#### 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。[【返回】](#)

#### 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。[【返回】](#)

#### 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，而且自动更新：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 [【TIMESTAMP字段类型初始化】](#) [【返回】](#)

#### 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。[【返回】](#)

### 1.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。

[【返回】](#)

### 2.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。

[【返回】](#)

### 3.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。[【返回】](#)

### 4.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。[【返回】](#)

### 5.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。[【返回】](#)

### 6.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。[【返回】](#)

### 7.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。[【返回】](#)

### 8.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。[【返回】](#)

### 9.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。[【返回】](#)

### 10.用UNION ALL代替UNION。

UNION ALL不需要对结果集再进行排序。[【返回】](#)

### 11.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。 [【返回】](#)

## 12.建议使用合理的分页方式以提高分页效率。

假如有类似下面面分页语句：

SELECT \* FROM table ORDER BY TIME DESC LIMIT 10000, 10; 这种分页方式会导致大量的io，因为MySQL使用的是提前读取策略。

推荐分页方式：

SELECT \* FROM table WHERE TIME<last\_TIME ORDER BY TIME DESC LIMIT 10.

SELECT \* FROM table inner JOIN (SELECT id FROM table ORDER BY TIME LIMIT 10000, 10) as t USING(id) [【返回】](#)

## 13.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。 [【返回】](#)

## 14.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。

[【返回】](#)

## 15.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。 [【返回】](#)

## 16.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() [【返回】](#)

## 17.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。 [【返回】](#)

## 18.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); [【返回】](#)

19.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。[【返回】](#)

20.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。[【返回】](#)

21.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。[【返回】](#)

22.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。[【返回】](#)

23.建表语句示例

```
CREATE TABLE `pay_notify_test` (  
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'id',  
  `gw_pay_id` varchar(32) NOT NULL COMMENT '网关支付流水',  
  `notice_firsttime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '第一次通知事件',  
  `notice_lasttime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '最后一次通知时间',  
  `notice_nexttime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '下一次通知时间',  
  `notice_count` tinyint(3) unsigned NOT NULL DEFAULT '0' COMMENT '通知次数,最大20次',  
  `notice_url` varchar(300) NOT NULL DEFAULT '' COMMENT '通知地址',  
  `notice_status` tinyint(4) NOT NULL DEFAULT '0' COMMENT '通知状态, 0: 未通知, 1: 正在通知, 2: 已通知',  
  `version` smallint(5) unsigned NOT NULL COMMENT '版本号',  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `uniq_gpid` (`gw_pay_id`),  
  KEY `idx_notice_nexttime` (`notice_nexttime`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT '支付通知';
```

[【返回】](#)

24.产品线对应名称说明(供参考)

事业部	命名简写	事业部	命名简写
机票	f	度假	vc
酒店	h	旅行	lv
无线	wap	门票	tk
兜行	dx	旅图	lv
搜索	s	ops	ops
支付平台	pay	dba	dba
qss	qss	火车票	t

事业部	命名简写	事业部	命名简写
特殊项目部	sp	呼叫中心	cc
用户中心	uc	市场部	mkt

[【返回】](#)