# mysql规范

## 基础规范

1. 【强制】必须使用InnoDB存储引擎

说明：支持事务、行级锁、并发性能更好、CPU及内存缓存页优化使得资源利用率更高

1. 【强制】必须使用UTF8字符集

说明：万国码，无需转码，无乱码风险，节省空间, 如果要使用表情，那么使用 utfmb4来进行存储，注意它与 utf-8编码的区别。

1. 【强制】数据表、数据字段必须加入中文注释
2. 【推荐】禁止主库使用存储过程、视图、触发器、Event

说明: 高并发大数据的游戏业务，架构设计思路是“解放数据库CPU，将计算转移到服务层”，并发量大的情况下，这些功能很可能将数据库拖死，业务逻辑放到服务层具备更好的扩展性，能够轻易实现“增机器就加性能”。数据库擅长存储

## 建表规范

1. 【强制】单表列数目必须小于50
2. 【强制】表必须有主键，例如自增主键

说明：

a）主键递增，数据行写入可以提高插入性能，可以避免page分裂，减少表碎片提升空间和内存的使用

b）主键要选择较短的数据类型， Innodb引擎普通索引都会保存主键的值，较短的数据类型可以有效的减少索引的磁盘空间，提高索引的缓存效率

c） 无主键的表删除，在row模式的主从架构，会导致备库夯住

1. 【强制】表名必须使用小写字母或者数字，表命名规范:用户表 t\_u\_xxxx,游戏配置数据表 t\_cfg\_xxxx,日志数据表 t\_log\_xxxx,公共数据表t\_pub\_xxxx。
2. 【强制】表达是与否概念的字段，必须使用 IsXxx的方式命名，数据类型是 unsignedtinyint（ 1表示是，0表示否），此规则同样适用于 odps建表。说明：任何字段如果为非负数，必须是 unsigned。
3. 【强制】禁止出现数字开头，单词首字母大小。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。

正例：GetterAdmin，TaskConfig，Level3Name反例：getter\_admin，taskConfig，level\_3\_name

1. 【强制】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。
2. 【强制】禁用保留字，如 desc、range、match、delayed等，请参考 MySQL官方保留字。
3. 【强制】唯一索引名为 uk\_字段名；普通索引名则为 idx\_字段名。

说明：uk\_ 即 unique key；idx\_ 即 index的简称。

1. 【强制】小数类型为 decimal，禁止使用 float和 double。

说明：float和 double在存储的时候，存在精度损失的问题，很可能在值的比较时，得到不正确的结果。如果存储的数据范围超过 decimal的范围，建议将数据拆成整数和小数分开存储。

1. 【强制】如果存储的字符串长度几乎相等，使用 char定长字符串类型。
2. 【强制】varchar是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 text，独立出来一张表，用主键来对应，避免影响其它字段索引效率。
3. 【推荐】表必备三字段：id, CreateDate, UpdateDate。

说明：其中 id必为主键，类型为 unsigned bigint、单表时自增、步长为 1。CreateDate,

UpdateDate的类型均为 date\_time类型。

1. 【推荐】表的命名最好是加上“业务名称\_表的作用”。

正例：tiger\_task / tiger\_reader / mpp\_config

1. 【推荐】库名与应用名称尽量一致。
2. 【推荐】如果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。
3. 【推荐】字段允许适当冗余，以提高性能，但是必须考虑数据同步的情况。冗余字段应遵循：

1）不是频繁修改的字段。

2）不是 varchar超长字段，更不能是 text字段。

正例：商品类目名称使用频率高，字段长度短，名称基本一成不变，可在相关联的表中冗余存储类目名称，避免关联查询。

1. 【推荐】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。

说明：如果预计三年后的数据量根本达不到这个级别，请不要在创建表时就分库分表。

1. 【参考】合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。

正例：人的年龄用 unsigned tinyint（表示范围 0-255，人的寿命不会超过 255岁）；海龟就必须是 smallint，但如果是太阳的年龄，就必须是 int；如果是所有恒星的年龄都加起来，那么就必须使用 bigint。

## 索引规范

1. 【强制】单表索引建议控制在5个以内。
2. 【强制】单索引字段数不允许超过5个。

说明：字段超过5个时，实际已经起不到有效过滤数据的作用了。

1. 【强制】禁止在更新十分频繁、区分度不高的属性上建立索引。

说明：

a）更新会变更B+树，更新频繁的字段建立索引会大大降低数据库性能

b）“性别”这种区分度不大的属性，建立索引是没有什么意义的，不能有效过滤数据，性能与全表扫描类似

1. 【强制】业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。

说明：不要以为唯一索引影响了 insert速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外，即使在应用层做了非常完善的校验和控制，只要没有唯一索引，根据墨菲定律，必然有脏数据产生。

1. 【强制】 超过三个表禁止 join。需要 join的字段，数据类型保持绝对一致；多表关联查询时，保证被关联的字段需要有索引。

说明：即使双表 join也要注意表索引、SQL性能。

1. 【强制】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。

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。

1. 【强制】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。

说明：索引文件具有 B-Tree的最左前缀匹配特性，如果左边的值未确定，那么无法使用此索引。

1. 【推荐】如果有 order by的场景，请注意利用索引的有序性。order by 最后的字段是组合索引的一部分，并且放在索引组合顺序的最后，避免出现 file\_sort的情况，影响查询性能。

正例：where a=? and b=? order by c; 索引：a\_b\_c

反例：索引中有范围查找，那么索引有序性无法利用，如：WHERE a>10 ORDER BY b; 索引a\_b无法排序。

1. 【推荐】利用覆盖索引来进行查询操作，来避免回表操作。

说明：如果一本书需要知道第 11章是什么标题，会翻开第 11章对应的那一页吗？目录浏览一下就好，这个目录就是起到覆盖索引的作用。

正例：能够建立索引的种类：主键索引、唯一索引、普通索引，而覆盖索引是一种查询的一种效果，用 explain的结果，extra列会出现：using index。

1. 【推荐】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.\* FROM 表 1 a, (select id from 表 1 where 条件 LIMIT 100000,20 ) b where a.id=b.id

1. 【推荐】SQL性能优化的目标：至少要达到 range 级别，要求是 ref级别，如果可以是 consts最好。

说明：

1）consts 单表中最多只有一个匹配行（主键或者唯一索引），在优化阶段即可读取到数据。

2）ref 指的是使用普通的索引（normal index）。

3）range 对索引进行范围检索。

反例：explain表的结果，type=index，索引物理文件全扫描，速度非常慢，这个 index级

别比较 range还低，与全表扫描是小巫见大巫。

1. 【推荐】建组合索引的时候，区分度最高的在最左边。

正例：如果 where a=? and b=? ，a列的几乎接近于唯一值，那么只需要单建 idx\_a索引即可。

说明：存在非等号和等号混合判断条件时，在建索引时，请把等号条件的列前置。如：where a>?

and b=? 那么即使 a的区分度更高，也必须把 b放在索引的最前列。

1. 【参考】创建索引时避免有如下极端误解：

1）误认为一个查询就需要建一个索引。

2）误认为索引会消耗空间、严重拖慢更新和新增速度。

3）误认为唯一索引一律需要在应用层通过“先查后插”方式解决。

## SQL规范

1. 【强制】禁止使用SELECT \*，只获取必要的字段，需要显示说明列属性

说明：

a）读取不需要的列会增加CPU、IO、NET消耗

b）不能有效的利用覆盖索引

c）使用SELECT \*容易在增加或者删除字段后出现程序BUG

1. 【强制】禁止使用INSERT INTO t\_xxx VALUES(xxx)，必须显示指定插入的列属性

说明：容易在增加或者删除字段后出现程序BUG

1. 【强制】禁止使用属性隐式转换

说明：SELECT uid FROM t\_user WHERE phone=13812345678 会导致全表扫描，而不能命中phone索引

1. 【强制】禁止在WHERE条件的属性上使用函数或者表达式

说明：SELECT uid FROM t\_user WHERE from\_unixtime(day)>='2017-02-15' 会导致全表扫描

正确的写法是：SELECT uid FROM t\_user WHERE day>= unix\_timestamp('2017-02-15 00:00:00')，使用函数或者表达式会导致索引失效。

1. 【强制】禁止负向查询，以及%开头的模糊查询

说明：

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

1. 【强制】不要使用 count(列名)或 count(常量)来替代 count(\*)，count(\*)就是 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。

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

1. 【强制】count(distinctcol) 计算该列除 NULL之外的不重复数量。注意 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。

1. 【强制】当某一列的值全是 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。

正例：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;

1. 【强制】使用 ISNULL()来判断是否为 NULL值。注意：NULL与任何值的直接比较都为 NULL。说明：

1） NULL<>NULL的返回结果是 NULL，而不是 false。

2） NULL=NULL的返回结果是 NULL，而不是 true。

3） NULL<>1的返回结果是 NULL，而不是 true。

1. 【强制】 在代码中写分页查询逻辑时，若 count为 0应直接返回，避免执行后面的分页语句。
2. 【强制】不得使用外键与级联，一切外键概念必须在应用层解决。

说明：（概念解释）学生表中的 student\_id是主键，那么成绩表中的 student\_id则为外键。如果更新学生表中的 student\_id，同时触发成绩表中的 student\_id更新，则为级联更新。外键与级联更新适用于单机低并发，不适合分布式、高并发集群；级联更新是强阻塞，存在数据库更新风暴的风险；外键影响数据库的插入速度。

1. 【强制】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。
2. 【强制】数据订正时，删除和修改记录时，要先 select，避免出现误删除，确认无误才能执行更新语句。
3. 【推荐】in操作能避免则避免，若实在避免不了，需要仔细评估 in后边的集合元素数量，控制在 1000个之内。
4. 【参考】如果有全球化需要，所有的字符存储与表示，均以 utf-8编码，那么字符计数方法

注意：

说明：

SELECT LENGTH("轻松工作")； 返回为 12

SELECT CHARACTER\_LENGTH("轻松工作")； 返回为 4

如果要使用表情，那么使用 utfmb4来进行存储，注意它与 utf-8编码的区别。

1. 【参考】TRUNCATETABLE 比 DELETE 速度快，且使用的系统和事务日志资源少，但 TRUNCATE

无事务且不触发 trigger，有可能造成事故，故不建议在开发代码中使用此语句。

说明：TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同。