一、数据库命令规范

* 所有数据库对象名称必须使用小写字母并用下划线分隔。
* 所有数据库对象名称禁止使用MySQL保留关键字（如果表名中包含关键字查询时，需要将其用单引号括起来）。
* 数据库对象的命名要能做到见名识意，并且最多不要超过32个字符。
* 临时库表必须以temp\_为前缀并以日期为后缀，备份表必须以bak\_为前缀并以日期（时间戳）为后缀。
* 所有存储相同数据的列名和列类型必须一致（一般作为关联列，如果查询时关联列类型不一致会自动进行数据类型隐式转换，会造成列上的索引失效，导致查询效率降低）

二、数据库基本设计规范

#### 1、所有表必须使用InnoDB存储引擎

没有特殊要求的情况下，所有表必须使用InnoDB存储引擎。

#### 2、数据库和表的字符集统一使用UTF-8

兼容性更好，统一字符集可以避免由于字符集转换产生的乱码，不同的字符集进行比较前需要进行转换会造成索引失效，如果数据库中有存储emoji表情的需要，字符集需要采用utf8mb4字符集。

#### 3、所有表和字段都需要添加注解

使用comment从句添加表和列的备注，从一开始就进行数据字典的维护。

#### 4、尽量控制单表数据量的大小，建议控制在500万以内

500万并不是MySQL数据库的限制，过大会造成修改表结构，备份，恢复都会有很大的问题。可以用历史数据归档（应用于日志数据），分库分表（应用于业务数据）等手段来控制数据量大小。

#### 5、谨慎使用MySQL分区表

* 分区表在物理上表现为多个文件，在逻辑上表现为一个表；
* 谨慎选择分区键，跨分区查询效率可能更低；
* 建议采用物理分表的方式管理大数据。

#### 6、尽量做到冷热数据分离，减少表的宽度

MySQL限制每个表最多存储4096列，并且每一行数据的大小不能超过65535字节。

* 减少磁盘I/O，保证数据的内存缓存命中率（表越宽，把表装载进内存缓冲池时所占用的内存也就越大，也会消耗更多的I/O）；
* 更有效的利用缓存，避免读入无用的冷数据；
* 经常一起使用的列放到一个表中（避免更多的关联操作）。

#### 7、禁止在表中建立预留字段

* 预留字段的命名很难做到见名识义；
* 预留字段无法确认存储的数据类型，所以无法选择合适的类型；
* 对预留字段类型的修改，会对表进行锁定。

#### 8、禁止在数据库存储图片，文件等大的二进制数据

通常文件很大，会短时间内造成数据量快速增长，数据库进行数据库读取时，通常会进行大量的随机I/O操作，耗时久。

通常存储于文件服务器，数据库只存储而文件地址信息。

#### 9、禁止在线上做数据库压力测试

#### 10、禁止从开发环境，测试环境直接连接生产环境数据库

三、数据库字段设计规范

#### 1、优先选择符合存储需要的最小的数据类型

原因：

列的字段越大，建立索引时所需要的空间也就越大，这样一页中所能存储的索引节点的数量也就越少，在遍历时所需要的I/O次数也就越多，索引的性能也就变差。

方法：

1. 将字符串转换成数字类型存储，如：将IP地址转换成整形数据。

MySQL提供了两个方法来处理ip地址：

* inet\_aton把ip转为无符号整形(4-8位)
* inet\_ntoa把整型的ip转为地址

插入数据前先用inet\_aton把ip地址转为整型，可以节省空间，显示数据时，使用inet\_ntoa把整型的ip转换为地址显示即可。

UPDATE vendors

SET ip\_address = INET\_ATON( '192.168.75.55' )

WHERE

vend\_id = 1001;

SELECT

INET\_NTOA( ip\_address )

FROM

vendors

WHERE

vend\_id = 1001;

1. 对于非负型的数据（如自增ID，整形IP）来说，要优先使用无符号整型来存储。

无符号相对于有符号可以多出一倍的存储空间。

SIGNED INT -2147483648~2147483647

UNSIGNED INT 0~4294967295

VARCHAR(N)中的N代表的是字符数，而不是字节数，使用UTF-8存储255个汉字Varchar(255)=765个字节。过大的长度会消耗更多的内存。

#### 2、避免使用TEXT,BLOB数据类型，最常见的TEXT类型可以存储64k的数据

* 建议把BLOB或是TEXT列分离到单独的扩展表中

MySQL 内存临时表不支持 TEXT、BLOB 这样的大数据类型，如果查询中包含这样的数据，在排序等操作时，就不能使用内存临时表，必须使用磁盘临时表进行。而且对于这种数据，MySQL 还是要进行二次查询，会使 sql 性能变得很差，但是不是说一定不能使用这样的数据类型。

如果一定要使用，建议把 BLOB 或是 TEXT 列分离到单独的扩展表中，查询时一定不要使用 select \* 而只需要取出必要的列，不需要 TEXT 列的数据时不要对该列进行查询。

* TEXT或BLOB类型只能使用前缀索引

MySQL对索引字段长度是有限制的，所以TEXT类型只能使用前缀索引，并

且TEXT列上是不能有默认值的。

#### 3、避免使用ENUM类型

* 修改ENUM值需要使用ALTER语句
* ENUM类型的ORDER BY操作效率低，需要额外操作
* 禁止使用数值作为ENUM的枚举值。

#### 4、尽可能把所有列定义为NOT NULL

原因：

* 索引NULL列需要额外的空间来保存，所以要占用更多的空间
* 进行比较和计算时要对NULL值做特别的处理

#### 5、使用TIMESTAMP（4个字节）或DATETIME类型（8个字节）存储时间

TIMESTAMP存储的时间范围1970-01-01 00:00:01~2038-01-19-03:14:07，占用4字节和INT相同，但比INT可读性高。

超出TIMESTAMP取值范围的使用DATATIME类型存储。

用字符串存储日期类型的数据。❌

* 无法用日期函数进行计算和比较
* 用字符串存储日期要占用更多的空间

#### 6、同财务相关的金额类数据必须使用decimal类型

* 非精准浮点：float，double
* 精准浮点：decimal

Decimal在计算时不会丢失精度，占用空间由定义的宽度决定，每4个字节可以存储9位数字，并且小数点要占用一个字节。可用于存储比bigint更大的整型数据。

四、索引设计规范

#### 1、限制每张表上的索引数量，建议单张表索引不超过5个

索引可以增加查询效率，但同样也会降低插入和更新的效率，甚至有些情况下会降低查询效率。

MySQL优化器在选择如何优化查询时，会根据统一信息，对每个可以用到的索引来进行评估，以生成出一个最好的执行计划，如果同时有很多个索引都可以用于查询，就会增加MySQL优化器生成执行计划的时间，同样会降低查询性能。

#### 2、禁止给表中的每一列都建立单独的索引

5.6 版本之前，一个 sql 只能使用到一个表中的一个索引，5.6 以后，虽然有了合并索引的优化方式，但是还是远远没有使用一个联合索引的查询方式好。

#### 3、每个InnoDB表必须有个主键

InnoDB是一种索引组织表，数据的存储的逻辑顺序和索引的顺序是相同的。每个表都可以有多个索引，但是表的存储顺序只能有一种。

InnoDB是按照主键索引的顺序来组织表的。

* 不要使用更新频繁的列作为主键，不适用多列主键（相当于联合索引）
* 不要使用UUID，MD5，HASH，字符串列作为主键（无法保证数据的顺序增长）
* 主键建议使用自增ID值

#### 4、常见索引列建议

* 出现在SELECT、UPDATE、DELETE语句的WHERE从句中的列。
* 包含在ORDER BY、GROUP BY、DISTINCT中的字段。
* 并不要将符合上述2条中的字段的列都建立一个索引，通常将1、2中的字段建立联合索引效果更好。
* 多表join的关联列。

#### 5、如何选择索引列的顺序

建立索引的目的是：希望通过索引进行数据查找，减少随机IO，增加查询性能，索引能过滤出越少的数据，则从磁盘中读入的数据也就越少。

* 区分度最高的放在联合索引的最左侧（区分度=列中不同值的数量/列的总行数）
* 尽量把字段长度小的列放在联合索引的最左侧（因为字段越小，一页能存储的数据量越大，IO性能也就越好）
* 使用最频繁的列放到联合索引的左侧（这样可以比较少的建立一些索引）

#### 6、避免建立冗余索引和重复索引

* 重复索引示例：primary key(id)、index(id)、unique index(id)
* 冗余索引示例：index(a,b,c)、index(a,b)、index(a)

#### 7、对于频繁的查询优先考虑使用覆盖索引

覆盖索引：就是包含了所有查询字段（where，select，order by，group by包含的字段）的索引。

好处：

* 避免InnoDB表进行索引的二次查询：InnoDB是以聚集索引的顺序来存储的，对于InnoDB来说，二级索引在叶子节点中所保存的是行的主键信息，如果是用二级索引查询数据的话，在查找到相应的键值后，还要通过主键进行二次查询才能获取我们真实所需要的数据。而在覆盖索引中，二级索引是按键值中可以获取所有的数据，避免了对主键的二次查询，减少了IO操作，提升了查询效率。
* 可以把随机IO变成顺序IO加快查询效率；由于覆盖索引是按键值的顺序存储的，对于IO密集型的范围查找来说，对比随机从磁盘读取每一行的数据IO要少的多，因此利用覆盖索引在访问时也可以把磁盘的随机读取IO转变成索引查找的顺序IO。

#### 8、索引SET规范

尽量避免使用外键约束

* 不建议使用外键约束（foreign key），但一定要在表与表之间的关联键上建立索引。
* 外键可用于保证数据的参照完整性，但建议在业务端实现。
* 外键会影响父表和字表的写操作从而降低性能。