## 创建表的注意事项

      字段名及字段配制合理性

1、        剔除关系不密切的字段

2、        字段命名要有规则及相对应的含义（不要一部分英文，一部分拼音，还有类似a.b.c这样不明含义的字段）

3、        字段命名尽量不要使用缩写（大多数缩写都不能明确字段含义）

4、        字段不要大小写混用（想要具有可读性，多个英文单词可使用下划线形式连接）

5、        字段名不要使用保留字或者关键字

6、        保持字段名和类型的一致性

7、        慎重选择数字类型

8、        给文本字段留足余量

      系统特殊字段处理及建成后建议

1、        添加删除标记（例如操作人、删除时间）

2、        建立版本机制

      表结构合理性配置

1、        多型字段的处理

就是表中是否存在字段能够分解成更小独立的几部分（例如：人可以分为男人和女人）

2、        多值字段的处理

可以将表分为三张表

这样使得检索和排序更加有调理，且保证数据的完整性！

其它建议

1、        对于大数据字段，独立表进行存储，以便影响性能（例如：简介字段）

2、        使用varchar类型代替char，因为varchar会动态分配长度，char指定长度是固定的。

3、        给表创建主键，对于没有主键的表，在查询和索引定义上有一定的影响。

4、        避免表字段运行为null，建议设置默认值（例如：int类型设置默认值为0）在索引查询上，效率立显！

5、        建立索引，最好建立在唯一和非空的字段上，建立太多的索引对后期插入、更新都存在一定的影响（考虑实际情况来创建）。

表与表之间的关系，字段之间也有联系，字段本身还有着自己的属性，一旦这些结合在一起，往往会出现一些意想不到的结果。

    1.字段属性与外键。外键是什么就不解释了，设表2中的字段B要关联表1中的字段A，那么A就必须为主键或具有Unique属性，其实关键还是在于字段A必须具有唯一性。道理很简单，就好像登录名不能重复一样，假设登录名重复，那么登录时很明显系统会出错，因为不知道是哪个用户，同理，字段A必须要唯一。

    2.还是外键。一般设置外键时默认约束是Restrict，而这其实是不好的，尤其是当你想修改一个表中的数据却发现结果失败的时候。而约束有Delete和Update时需要注意其中的SET NULL，还是以表1、2为例，如果字段A是NOT NULL时，设置SET NULL是会失败的。道理很明显，不解释了。

# MYSQL建表优化考虑事项

# 1.  char与varchar

char ：长度固定，比较适合存储很短（比如门牌号码101，201）、固定长度（比如使用uuid作为主键）、十分频繁改变的column的字段；char(M)类型的数据列里，每个值都占用M个字节，如果某个长度小于M，MySQL就会在它的右边用空格字符补足。（在检索操作中那些填补出来的空格字符将被去掉）

varchar：可变长度，占用长度为字符数+1(用来存储位置)   
总结：char 因固定长度，所以在处理速度上要比varchar快速很多，但是相对较费存储空间；所以对存储空间要求不大，但在速度上有要求的可以使用char类型，反之可以用varchar类型来实例

# 2.   存储引擎

|  |  |  |
| --- | --- | --- |
|  | MyISAM | InnoDB |
| 大概比较 | 不是事务安全的，也不支持外键。如果事物回滚将造成不完全回滚，不具有原子性。如果执行大量的SELECT，MyISAM是更好的选择。 | 事务安全的、支持外键。你的数据执行大量的INSERT或UPDATE,出于性能方面的考虑，应该使用InnoDB表 |
| 数据占用 | MyISAM的索引和数据是分开的，并且索引是有压缩的，内存使用率就对应提高了不少。能加载更多索引。表格可以被压缩,而且它们支持全文搜索。 | Innodb是索引和数据是紧密捆绑的，没有使用压缩从而会造成Innodb比MyISAM体积庞大不小。 |
| 读写性能 | 读性能较强 | 写性能较强 |
| 锁 | 整表锁。  对于select count(\*) from table 的操作，MyISAM有单独保存表行数，不需读表，可以直接读取。 | 行级锁。  不保存表的具体行数，所以select count(\*) from table 是全表扫描。但是当count(\*)语句包含 where条件时与MyISAM没有区别。 |
| Char&varchar | 建议使用固定长度的数据列代替可变长的数据列。缺点就是占用磁盘空间。 | 建议使用varchar类型。对于innodb数据表，内部的行存储格式没有区分固定长度和可变长度列(所有数据行都使用指向数据列值的头指针)，因此在本质上，使用固定长度的char列不一定比使用可变长度varchar的性能要好，因此，主要的性能因素是数据行使用的存储总量，由于char平均占用的空间多于varchar，因此使用varchar来最小化需要处理的数据行的存储总量和磁盘I/O是比较好的。 |

# 3.  列类型转换规则

在MySQL中用来判断是否需要进行对据列类型转换的规则

１、在一个数据表里，如果每一个数据列的长度都是固定的，那么每一个数据行的长度也将是固定的．

２、只要数据表里有一个数据列的长度的可变的，那么各数据行的长度都是可变的．

３、如果某个数据表里的数据行的长度是可变的，那么，为了节约存储空间，MySQL会把这个数据表里的固定长度类型的数据列转换为相应的可变长度类型．但长度小于4个字符的char数据列不会被转换为varchar类型

4、导致实际应用中varchar长度限制的是一个行定义的长度。 MySQL要求一个行的定义长度不能超过65535。

# 4.   主键、外键

主键：尽可能使用长度短的主键，如果可以使用外键做主键则更好。在主键上无需建单独的索引，因为系统内部为主键建立了聚簇索引。

外键：外键会影响插入和更新性能，对于批量可靠数据的插入，建议先屏蔽外键检查。

      对于数据量大的表，建议去掉外键，改由应用程序进行数据完整性检查。

尽可能用选用对应主表的主键作作为外键，避免选择长度很大的主表唯一键作为外键。

外键是默认加上索引的。

# 5.   字段

1.选用字段长度最小

2. 优先使用定长型

3.尽可能的定义 “NOT NULL”

4.数值型字段中避免使用 “ZEROFILL”

5.如果要储存的数据为字符串,且可能值已知且有限, 优先使用 enum 或 set

# 6.   索引

对于那些在查询中很少使用或者参考的列不应该创建索引。费空间

对于那些只有很少数据值的列也不应该增加索引。映射太少

对于那些定义为text, image和bit数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少。

当修改性能远远大于检索性能时，不应该创建索引。这是因为，修改性能和检索性能是互相矛盾的。

1.   被索引的字段的长度越小,该索引的效率越高

2.   被索引的字段中,值的重复越少, 该索引的效率越高

3.   查询语句中,如果使用了 “group”子句, 根据其中字段出现的先后顺序建立多字段索引

4.   查询语句中,如果使用了 “distinct”,根据其中字段出现的先后顺序建立多字段索引

5.   “where”子句中, 出现对同一表中多个不同字段的 “and”条件时, 按照字段出现的先后顺序建立多字段索引

6.   “where”子句中, 出现对同一表中多个不同字段的 “or”条件时, 对重复值最少的字段建立单字段索引

7.   进行 “内/外连接”查询时, 对 “连接字段”建立索引

8.   对 “主键”的 “unique” 索引毫无意义,不要使用。对于一个Primary Key的列，MySQL已经自动对其建立了Unique Index，无需重复再在上面建立索引了。

9.   被索引字段尽可能的使用 “NOT NULL”属性

10. 对写入密集型表,尽量减少索引, 尤其是 “多字段索引”和 “unique” 索引

11. MySQL只会使用前缀，例如key(a, b) …where b=5将使用不到索引。

12. 控制单个索引的长度。使用key(name(8))在数据的前面几个字符建立索引

13. 相近的键值比随机好。Auto\_increment就比uuid好。

# 7.   查询语句的优化

1.   多多利用 “explain”查询索引使用情况, 以便找出最佳的查询语句写法和索引设置方案

2.   慎用 “select \*”,查询时只选出必须字段

3.   查询使用索引时,所遍历的索引条数越少,索引字段长度越小, 查询效率越高 (可使用 “explain”查询索引使用情况)

4.   避免使用 mysql函数对查询结果进行处理,将这些处理交给客户端程序负责

5.   使用 “limit”时候, 尽量使 “limit” 出的部分位于整个结果集的前部, 这样的查询速度更快, 系统资源开销更低

6.   在 “where”子句中使用多个字段的 “and”条件时, 各个字段出现的先后顺序要与多字段索引中的顺序相符

7.   在 “where”子句中使用 “like”时, 只有当通配符不出现在条件的最左端时才会使用索引

8.   在 mysql 4.1以上版本中, 避免使用子查询, 尽量使用 “内/外连接”实现此功能

9.   减少函数的使用,如果可能的话, 尽量用单纯的表达式来代替

10.  避免在 “where”子句中, 对不同字段进行 “or” 条件查询, 将其拆分成多个单一字段的查询语句效率更高

11. 查询时使用匹配的类型。例如select \* from awhere id=5，如果这里id是字符类型，同时有index，这条查询则使用不到index，会做全表扫描，速度会很慢。正确的应该是… where id=”5” ，加上引号表明类型是字符。