1. 对主键建立索引（主键具有唯一性，适合建立索引，而且外键关联查找的时候需要用到）。

对查询操作频繁，写入操作较少的列制作索引。比如book表中的title、isbn，book表中的description也可以制作索引，但由于description数据量很大，索引生成的B+tree占用存储空间很大，因此可以选取前面十个字节制作一个前缀索引，也可以将全文进行分词然后制作一个反向索引（之前已经用lucene实现过）。

对经常需要排序的列制作索引。比如order\_info表中的order\_date。

对可以用于分类的列建立索引，比如book表中的type，这样用户可以根据书籍类型查询书籍；又比如cart表中的userId，因为每次都是筛选某一用户的购物车数据返回给客户端。

同时需要注意，数据量很小的时候建立索引没有意义（比如我寥寥无几的书籍数据…）；不断有数据更新的表不适合建立太多索引，比如order\_info表一直有新的订单插入，cart表一直有增删改操作；有太多数据为NULL的字段也不适合建立索引。

1. Book表、user表的主键可以使用整数自增主键，order、cart表的主键可以使用UUID。因为book、user新数据插入不会特别快，而订单、购物车项的新数据非常多，uuid可以保证大量新数据插入时主键不重复。

Book\_description等包含大段文字的字段可以使用text或blob类型，将字段内容单独存储成文件，在数据表中只需要存储指针，大大节省存储空间。

Book\_title、book\_author等字段，长度变化比较显著，可以使用动态字符varchar，避免存储空间被不必要地浪费掉。

Book\_price使用decimal类型，因为考虑到实际意义，定价需要使用到小数。

Book\_isbn使用bigint类型，因为isbn是统一位数的一串数字，且用数字类型存储占用的空间比字符类型要少。

User\_type、user\_state字段虽然目前都只有两种类型，但考虑到之后可能会新增其他类型，所以使用tinyint类型。

其他字段基本都可以按照实际意义来选择类型。

1. 不需要另存一张表，多个邮箱就类似order\_item一样，需要分开存储并用外键关联，否则会产生大量冗余数据，但是只有一个邮箱，且邮箱属于较为常用的字段，所以直接存储在user表中即可。
2. 自增主键的优点：由数据库自动编号，速度快且不用人为考虑主键是否会重复导致新数据插入失败的问题；使用的是数字类型，较为简单；占用存储空间小；按顺序增长，有利于检索，可以知道数据插入的前后顺序。

自增主键的缺点：合并两张表中的数据时，可能会与另一张表中已有的主键重复，产生冲突。

UUID的优点：大量数据插入时不用担心主键重复（比如集群情况下）。

UUID的缺点：占用存储空间大；没有规律，无法区分数据插入的前后顺序。

1. MyISAM默认事务不安全所以不支持事务机制，InnoDB支持事物机制。

MyISAM不支持外键，InnoDB支持外键。

InnoDB可以实现更细粒度的锁（行锁），MyISAM最多锁到表。

MyISAM执行select语句要远快于InnoDB，因为MyISAM保存了表的具体行数，执行select语句时不需要像InnoDB一样扫描全表。但InnoDB执行update和insert语句要快于MyISAM。

InnoDB 是聚集索引，MyISAM 是非聚集索引。聚簇索引的文件存放在主键索引的叶子节点上，因此 InnoDB 必须要有主键，通过主键索引效率很高。但是辅助索引需要两次查询，先查询到主键，然后再通过主键查询到数据。因此，主键不应该过大，因为主键太大，其他索引也都会很大。而 MyISAM 是非聚集索引，数据文件是分离的，索引保存的是数据文件的指针。主键索引和辅助索引是独立的。

MyISAM支持支持三种不同的存储格式：静态表(默认，但是注意数据末尾不能有空格，会被去掉)、动态表、压缩表。当表在创建之后并导入数据之后，不会再进行修改操作，可以使用压缩表，极大的减少磁盘的空间占用。InnoDB需要更多的内存和存储，它会在主内存中建立其专用的缓冲池用于高速缓冲数据和索引。

图形用户界面

描述已自动生成

519021910721-陆伊敏