1. 数据库的相关知识和数据库的设计。

减少数据冗余，避免数据维护异常，节约储存空间，高效的访问

**设计步骤**

需求分析：数据是什么，有哪些属性，数据和属性的特点（存储特点），数据的

生命周期

逻辑设计：使用ER图对数据库进行逻辑设计

物理设计：把逻辑设计转成物理设计

维护优化：新的需求进行建表，索引优化，大表拆分

**需求分析**

1.       实体及实体之间的关系（1:1 1:n n:n）

2.       实体所包含的属性

3.       那些属性或属性的组合可以唯一标识一个实体

**实例**(以一个小型电子商务网站为例。用户模块，商品模块，订单模块，购

物车模块，供应商模块

用户模块：记录注册用户信息。唯一标示-用户名，身份证，电话。储

存特点，增长，永久储存你。

商品模块：…….，可以对下线商品归档存储（不能删除因为涉及订单

等，可以迁移）。

订单模块：………，永久存储（分表，分库存储）

购物车模块：……….，不用永久存储（设置归档，清理规则）

供应商模块：…….，永久存储

关系：)

**逻辑设计**

1.       将需求转化为数据库的逻辑模型

2.       通过ER图的型式对逻辑模型进行展示

3.       同所选用的具体的DBMS系统无关

**物理设计—设计标的结构**

1. 选择合适的数据库管理系统

[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)管理[系统](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F)(Database Management System)是一种操纵和管理数据库的大型软件，用于建立、使用和维护[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，简称[DBMS](https://baike.baidu.com/item/DBMS)。它对[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)进行统一的管理和[控制](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6/948689)，以保证[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)的安全性和完整性。用户通过[DBMS](https://baike.baidu.com/item/DBMS)访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)中的数据，[数据库管理员](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E5%91%98)也通过[dbms](https://baike.baidu.com/item/dbms)进行数据库的维护工作。它可使多个[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)和用户用不同的方法在同时或不同时刻去建立，修改和询问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)。大部分[DBMS](https://baike.baidu.com/item/DBMS)提供[数据定义语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%9A%E4%B9%89%E8%AF%AD%E8%A8%80)[DDL](https://baike.baidu.com/item/DDL/21997)（Data Definition Language）和[数据操作语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%93%8D%E4%BD%9C%E8%AF%AD%E8%A8%80)[DML](https://baike.baidu.com/item/DML)（Data Manipulation Language），供用户定义[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)的模式结构与权限约束，实现对数据的追加、删除等操作。

1. 定义数据库，表及字段的命名规范

**表及字段的命名规则**

1. 可读性原则 （驼峰命名，但注意表名的大小写敏感）

**1、小驼峰式命名法（lower camel case）：**

第一个单字以小写字母开始，第二个单字的首字母大写。例如：firstName、lastName。

**2、大驼峰式命名法（upper camel case）：**

每一个单字的首字母都采用大写字母，例如：FirstName、LastName、[CamelCase](https://baike.baidu.com/item/CamelCase/636859)，也被称为 Pascal 命名法。

变种：**StudlyCaps**，是“驼峰式大小写”的变种。

1. 表意性原则 表明对象、数据内容、存储过程

存储过程（Stored Procedure）是在大型[数据库系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F)中，一组为了完成特定功能的SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。

3.       长名原则 尽可能少用或不用缩写

3.       根据所选的DBMS系统选择合适的字段类型   varchar/char如何选择等

4.       反范式化设计 即增加冗余，用空间换取效率的提升

**字段类型的选择原则**

数据类型印象存储空间开销和查询性能。可以选择多数据类型时，优先数字，其次日期或二进制，最后字符串。同级别数据类型，优先占空间小的。

1.       同样的数据进行比较时（查询条件，join及排序）时，字符处理比数字慢。

2.       在数据库中，数据处理以页为单位，列的长度越小，利于性能提升。

Char和varchar如何选择

1.       数据长度差不多的选择char，数据长度不一致的时候varchar。

2.       列的最大数据长度小于50Byte，一般用char，但是如果这个列很少用，基于节省空间和减少I/O的考虑，可以选择varchar。

3.       一般不宜定义大于50Byte的char类型。

Decimal和float

1.       decimal用于存储精确数据，float只能用于存储非精确数据。

2.       float存储空间开销一般小于decimal（7位小数要4个字节，15位小数要8个字节）

时间类型

1.       int 字段长度比datetime小，但是用不方便需要函数转换，只能存储到2038-1-19 11:14:07。在不经常查询时间的时候用int。

2.       存储的时间粒度 年月日小时分秒周

**如何选择主键**

1.       业务主键用于表示业务数据，进行表与标的之间的关联。数据库主键为了优化数据存储（Innodb不手动设主键会自动生成6字节隐含主键）。

2.       根据数据库的类型，考虑主键是否要顺序增长。

3.       主键的字段类型所占空间要尽可能的小，对于使用聚集索引方式存储的表，每个索引后都会附件主键信息。

**避免使用外键约束**

1.       虽然保持数据完整性，但是降低数据导入的效率。（高并发的时候每次查询都会检测是否符合外检）

2.       降低维护成本。

3.       虽然不建议使用外键，但是相关联的列上一样要建立索引。

**维护优化**

1.       维护数据字典 要记住字典中每个数字代表什么含义 可以使用第三方工具也可以使用备注字段来维护 comment’ ’ 。

2.       维护索引 where groupby orderby的从句 可选择性高的列要放到索引的前面 不要包括太长的数据类型 过多的索引会降低亵渎效率 定期维护索引碎片 sql语句中不要使用强制索引关键字。

3.       维维护表结构 列的变更 控制表的宽度和大小（拆分）。

4.       在适当的时候对表进行水平拆分（表结构相同，Hash key）或垂直拆分（中间切分表，表结构不同）。

**适合的操作**

批量操作（不适合逐条操作）

禁止使用select \*查询（I/O浪费，表结构改了数据错误）

控制使用用户自定义函数（影响索引 使用函数则列中索引失效）

不要使用数据库中的全文索引（维护高，不适合中文）

***表的设计具体注意的问题：***

        1、数据行的**长度不要超过8020字节**，如果超过这个长度的话在物理页中这条数据会占用两行从而造成存储碎片，降低查询效率。

          2、能够用数字类型的字段**尽量选择数字类型**而不用字符串类型的（电话号码），这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接回逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

        3、对于不可变字符类型char和可变字符类型varchar 都是8000字节,char查询快，但是耗存储空间，varchar查询相对慢一些但是节省存储空间。在设计字段的时候可以灵活选择，例如用户名、密码等**长度变化不大的字段可以选择CHAR，对于评论等长度变化大的字段可以选择VARCHAR**。

        4、**字段的长度在最大限度的满足可能的需要的前提下，应该尽可能的设得短一些**，这样可以提高查询的效率，而且在建立索引的时候也可以减少资源的消耗。

2.可能会遇到的问题以及解决办法。

(1)在考虑整个系统的流程的时候，我们必须要考虑，在高并发大数据量的访问情况下，我们的系统会不会出现极端的情况。（例：对外统计系统在7月16日出现的数据异常的情况，并发大数据量的的访问造成，数据库的响应时间不能跟上数据刷新的速度造成。具体情况是：在日期临界时（00：00：00），判断数据库中是否有当前日期的记录，没有则插入一条当前日期的记录。在低并发访问的情况下，不会发生问题，但是当日期临界时的访问量相当大的时候，在做这一判断的时候，会出现多次条件成立，则数据库里会被插入多条当前日期的记录，从而造成数据错误），**数据库的模型确定下来之后，我们有必要做一个系统内数据流向图，分析可能出现的瓶颈。**

(2) 为了保证数据库的一致性和完整性，在逻辑设计的时候往往会设计过多的表间关联，尽可能的降低数据的冗余。（例:用户表的地区，我们可以把地区另外存放到一个地区表中）如果**数据冗余低，数据的完整性容易得到保证，提高了数据吞吐速度，保证了数据的完整性，清楚地表达数据元素之间的关系。**而对于多表之间的关联查询（尤其是大数据表）时，其性能将会降低，同时也提高了客户端程序的编程难度，因此，物理设计需折衷考虑，根据**业务规则**，确定对**关联表的数据量大小**、**数据项的访问频度**，对此类数据表**频繁的关联查询**应适当提高数据冗余设计但增加了表间连接查询的操作，也使得程序的变得复杂，为了提高系统的响应时间，合理的数据冗余也是必要的。设计人员在设计阶段应根据系统操作的类型、频度加以均衡考虑。

(3) **最好不要用自增属性字段作为主键与子表关联。不便于系统的迁移和数据恢复。对外统计系统映射关系丢失**。

(4) 使用无效的名称:我们用“order”命名了一张表。不过，或许你还记得，“order”在SQL中是保留字! 不要使用关键字来当做对象名称

1. 封号、视频是否允许观看的审核机制是如制定的。

封号：在用户出现违规现象时，一次，两次，可以进行短期封号，但若多次出现违规现象的话，就进行永久封号。

视频网站的视频内容审核需要三审（就是一审、二审、三审）分别对应不同的监管和控制标准。  
其次，题主提到的的机器审核，就职过的视频网站都有这个功能。  
具体原理是通过给视频文件创建指纹库来完成自动完成初步审核。若指纹库中没有相关的内容就走上面说的一、二、三了。我们知道的基本上是这个流程。

1. 空闲时间自学的知识（也可以是复习）