1.1
缺点一：建立、维护、使用数据库系统需要更多的知识、资金、技巧和时间。

缺点二：数据库的复杂性可能导致低性能。

1.4
1. 数据冗余和不一致：若视频资料依照其类型进行存储，一个视频可能同时归类于音乐和游戏类别，导致同一视频数据在多个位置被存储，从而产生信息重复和不一致性问题。
2. 数据访问困难：若组织需依据特定搜索条件（非仅关键词搜索）来查找视频资料，并且元数据存储在文件中，那么在不开发应用程序的情况下，检索相关数据将变得困难。
3. 数据孤立：若实施了权限控制，可能会遇到并发更新权限信息的问题。例如，一个文件的访问权限被更新为允许，而另一个文件却未同步更新，因为文件系统不支持原子性的事务处理。
4. 完整性难题：若需对用户名称或 ID 设置新的限制条件，将面临较大挑战。
5. 原子性难题：在上传视频时，应确保视频及视频元数据的添加是原子性的，否则可能导致数据不一致。此外，需要一个底层的恢复机制来保证在发生故障时仍能保持原子性。
6. 并发访问异常：如果两个用户同时关注同一个视频博主，可能会导致该博主的粉丝数量仅增加一次。
7. 安全性问题：用户个人信息泄露的风险较高。

1.5
web 中使用的查询是通过提供没有特定语法的关键字列表来指定的，结果通常是一个有序的 URL 列表，以及有关 URL 内容的信息片段。相反，数据库查询具有特定的语法，允许指定复杂的查询，且查询的结果总是一个表。

1.6
QQ、微信、steam、无畏契约

1.8
物理数据独立性是指数据库的物理存储方式与应用程序之间的分离，这意味着如果数据库的存储方式发生改变，应用程序和数据库的逻辑设计不需要随之调整。这种特性的价值在于它减少了系统间的相互依赖，提升了系统的适应性，增强了维护的便捷性，并有助于系统的优化工作。

1.11
并发控制管理器

1.12
在两层体系结构中，应用程序驻留在客户机上，通过查询语言表达式来调用服务器上的数据库系统功能。
而在一个三层体系结构中，客户机只作为一个前端并且不包含任何直接的数据库调用。客户端通常通过一个表单界面与应用服务器进行通信。而应用服务器与数据库系统通信与访问数据。
对 Web 应用来说，显然使三层体系结构更好。因为 Web 应用的访问量很大，客户机直接通

过查询语言与数据库系统进行交互可能会出现阻塞（访问量太大），数据更新不及时（高并发引起），数据丢失（大数据量）等问题，通过一个应用服务器，我们可以进行负载均衡、分发等设置，由此来缓解数据库系统的压力。

1.13

1.Auto-admin

2.map-reduce 程序设计框架

1.14

NoSQL 系统在 21 世纪头十年出现的原因主要为：现代应用程序需要低延迟的读写速度，且随着大型应用的出现，需要 PB 级别的数据处理能力和百万级别的并发流量支持，系统管理员希望能够更简单地管理和部署分布式应用，而硬件、软件和人力成本上都能需要降低。

对比：

扩展性方面：

NoSQL：设计上支持简单的水平扩展，例如通过添加节点来增加容量。

传统数据库：通常难以扩展，特别是在需要执行复杂的多表连接时。

读写速度方面：

NoSQL：如 Redis 等实现快速读写操作，具备良好的性能。

传统数据库：随着数据量增加，复杂的逻辑可能导致性能下降，容易出现死锁等问题。

成本方面：

NoSQL：大多数 NoSQL 解决方案为开源，没有昂贵的许可费用。

传统数据库：企业级数据库的许可费用较高，并且随着系统规模的扩大而增加。

支持的功能方面：

NoSQL：通常不支持 SQL 标准，功能相对有限，大多数产品不支持事务等高级功能。

传统数据库：通常提供丰富的功能和特性，例如支持复杂查询、事务处理等。

1.15

1. 用户表：uid，用户名，密码的哈希值，联系方式（手机号、邮箱地址），头像的 url，个人简介
2. 好友关系表：另一个用户的 uid，建立好友关系的时间，当前的好友状态
3. 动态表：动态 id，发表动态的用户的 uid，动态的内容，动态的的点赞数，动态的评论数，动态的可见性