- DB
  - SQL
    - Data Manipulation Language
    - Data Definition Language
    - Data Control Language
  - Data Model
    - Relational
      - 对于many-to-one需要范式化，本身是个tradeoff
      - 不能很好地应对one-to-many的情况
    - Document
      - 适用于one-to-many的情况
      - schema不严格
      - 无法很好解决many-to-one的删除问题
  - 演化
    - NoSQL
      - 弱化transaction
      - 简化模型，使用KVS
      - 异步备份，不等待backup完成复制，而是primary完成就返回
      - 本质上用可以容忍的错误率换取了极大的性能提升
    - NewSQL
      - 对外提供SQL
      - 支持事务ACID
      - 并发控制不使用锁(否则太慢)
      - scale out & share nothing
      - 支持多机
  - DBMS
    - Buffer Pool
      - Storage
        - Heap File(使用链表或者目录来保存free pages)
        - Page(DB中的最小分配单元) 使用Slotted Page
        - Tuple(存储属性，对应于表的一行)
      - Frame(存放缓存的page)
      - Page Table(将page id映射到Frame)
      - 使用Latch处理并发的情况
      - 使用bitmap或者linked-list存放empty frame的信息
      - Evict策略
        - LRU以及近似的Clock Algorithm
          - 存在的问题：对于select *这样的搜索容易造成flooding
        - MRU LRU-K
      - 优化
        - Multi Buffer Pool
          - 每个page的访问模式是不一样的，将index和data分开存储
          - 避免高并发情况下的资源竞争，尤其是对latch的竞争
          - index需要经常访问，单个buffer pool很容易被覆盖
          - 可以使用hash或者range进行分配
        - 预取(Prefetching)
          - 虽然OS自带预取功能，但是OS的预取的大小可能和DB不一致
          - OS的预取基于自身文件系统的存储，不能很好适应DB的结构(包含更多语义，B+树)
        - Scan Sharing
          - 对于相邻的两条指令可以共享scan(30%的概率出现这样的情况)
        - 解决flooding问题
          - 使用多buffer pool，对于select *这样的问题就选择一个很小的buffer pool
          - 更换evict策略，使用MRU或者LRU(k)
          - IBM informix使用light scan，只使用一块小内存滑动窗口
      - Query模式优化
        - 上层的Index数据更容易被访问到，优化缓存
      - Evict成本
        - 对于非dirty page，直接evict即可
        - 对于dirty page，必须要写入硬盘，使用background writing，待系统空闲时写入
        - 使用WAL保证一致性
    - Query
      - 查询语句会被解析为一棵树
      - 处理模型
        - Iterator Model
          - 优点：易于实现
          - 缺点：使用太多的函数调用(尤其是C++中有大量的虚函数)，overhead特别大
        - Materialize Model
          - 优点：减少函数调用，适合OLTP，数量有限，效率高
          - 缺点：结果可能会特别大(可以通过LIMIT进行解决)，因而不适合OLAP(通常需要分析所有的数据)
        - Vectorized Model
          - 结合上面两者，适用于OLAP，批量处理
          - 相比于Iterator Model而言减少了函数调用
          - 相比于Materialize Model而言不至于一次性拿出所有的数据
      - 优化
        - Select优化
          - 对于相同的index，查询顺序可以影响查询效率
          - 使用bitmap进行集合运算