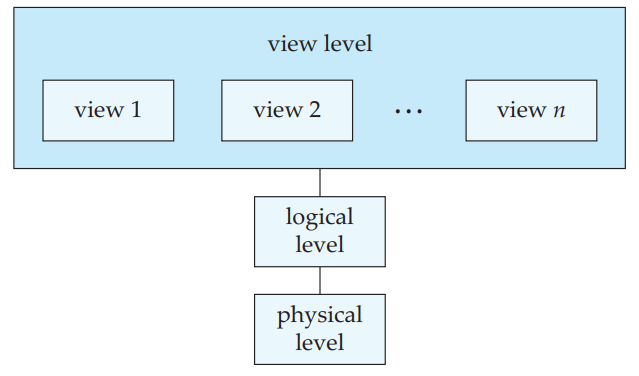
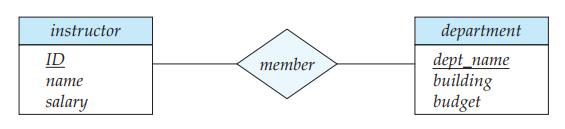
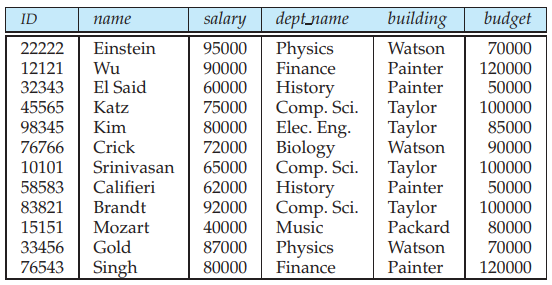
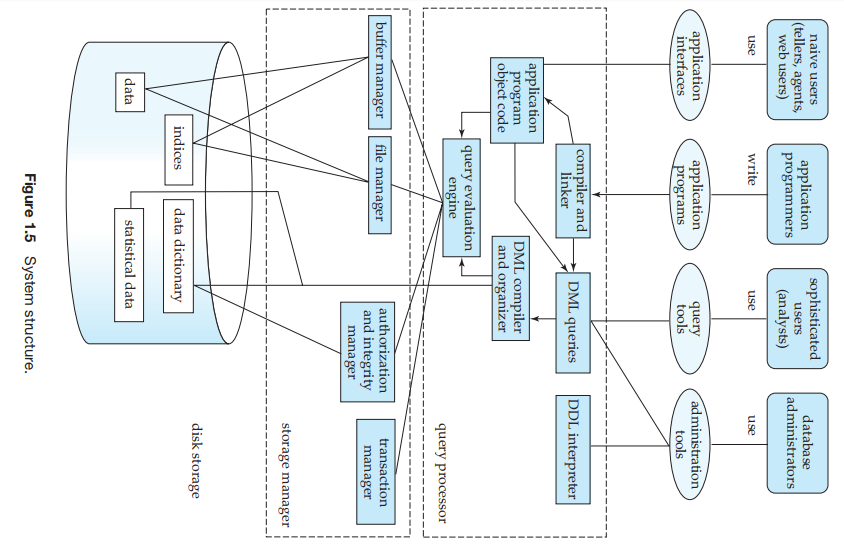
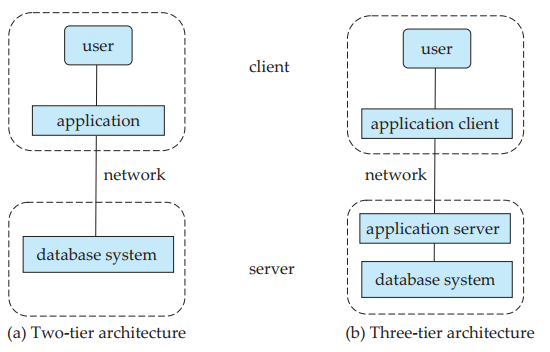
1. Introduction  
   DBMS(database-management system)包括数据库和用于存取数据的程序，DBMS的基本目标是为数据的存储提供方便、高效的方式，此外对大多数企业来说，数据是非常重要的,所以DBMS还要保证数据的安全，比如在系统奔溃或收到未授权访问请求的时候。
   1. Database-System Application
   2. Purpose of Database Systems  
      基于file-processing system的数据系统的问题：数据的丰富、一致性问题、不易维护、独立性、完整性、原子性、并发、安全问题
   3. View of Data  
      数据库系统隐藏了数据在存储、维护方面的细节，但是以视图的形式为用户提供数据。
      1. Data Abstraction 数据抽象  
         数据库系统必须能满足高效地查询数据的需求，为此设计者将数据库系统抽象为三层：物理层、逻辑层、视图层  
         
         1. 物理层，描述了数据在物理存储介质的实际存储方式
         2. 逻辑层，描述了数据在数据库的存储方式以及数据之间的逻辑关系，尽管为了实现这些逻辑关系需要复杂的物理关系，逻辑层为用户隐藏了这些复杂性。
         3. 视图层，这是对数据库最高级别的抽象，虽然逻辑层隐藏了物理层的复杂度，但对于一个完整的数据库所储存的信息来说，逻辑层仍然是非常复杂的，除了DBA有时可能直接操作逻辑层，普通用户往往只需要访问其中的一部分数据，视图层就是为了简化普通用户与数据库的交互而存在的。同一个数据库可以有多份视图。
      2. Instance and Schemas  
         随着数据的增删改，数据库会不断变化。在某一刻数据库中所有信息的集合称为Instance；而数据库的整体结构称为Schema，Schema很少发生变化。用程序来类比的话，Schema相当于预先声明的变量，Instance则相当于变量的值。  
         数据库有不同的分层，对应就有不同的Schema，视图层甚至有多个subschema。
      3. 数据模型  
         数据模型是用于描述数据、数据关系、数据语义、一致性约束的一系列工具的集合。数据模型可以从物理、逻辑、视图的层面描述数据库的设计，有多种类型：
         1. Relational Model  
            关系模型用表来描述数据及相互关系，每张表有多个列，每个列有唯一的名称，表和表之间又存在一定的关系。关系模型是一种基于记录的模型，采用这种模型记录的数据都有固定数量的字段或属性（列）。这种模型使用最广泛。
         2. Entity-Relationship Model  
            E-R模型用了实体和实体间关系这样的概念，实体对应与现实世界的对象。这种模型也被广泛使用。
         3. Object-Based Data Model  
            由于面向对象的编程语言成为主流，基于对象的数据模型也应运而生，这种模型结合和关系模型和E-R模型的特性，并增加了封装、方法等面向对象的思想。
         4. Semistructured Data Model  
            在XML等半结构化的数据模型中，允许数据有不同的字段，这与之前的模型都不相同。
   4. 数据库语言  
      SQL包含DDL（data-definition language）和DML（data-manpulate language）。
      1. DML  
         DML用于对数据的操作如增删改查，分为过程化DML和非过程化DML，前者在使用时要指定需要操作的数据以及操作的方式，后者只需要指定需要的数据就可以了，剩下的交给数据库系统（将查询语句翻译成对物理层的操作），所以后者更容易学习和使用。
      2. DDL  
         DDL用于定义数据的schema、存取方式、约束等。  
         域约束(domain constraints)：对数据类型的约束，每次插入数据都会检查  
         参照完整性(referential ingegrity)：要保证被关联的对象是存在的，比如course表中dept\_name与department表的某列关联，那么dept\_name的值必须是department实际存在的。数据库的修改有可能会破坏参照完整性，这时应该阻止操作以避免破坏参照完整性。  
         断言(assertions)：是数据库在任何时候都必须满足的条件，域约束和参照完整性是特殊的断言。一般的对类似“每个部门必须提供至少5个节目”这样的限制就可以创建断言，然后每次对数据改动前都会检查是否满足断言。  
         权限(authorization)：常见的权限类型有读取、插入、更新、删除等，可以控制用户拥有某几项权限。  
         DDL语句执行的结果为数据字典，这种数据字典可以看作是只能被数据库系统维护的特殊表。数据库在进行数据的查询、操作前都会先查阅数据字典。
   5. 关系型数据库
      1. 表
      2. DML  
         SQL语言是非过程的，一次查询中，可以输入多张表，但结果只是一张表。比如  
         select instructor.name   
         from instructor   
         where instructor.dept\_name=’history’，  
         查询语句定义了要查找dept\_name列为history的数据，并显示name列，执行后得到的是一张表，只有一个name列，每个符合dept\_name=’history’筛选条件的数据作为一行。
      3. DDL  
         DDL可以定义表、约束、断言等，比如有如下创建表的语句：  
         create table department(  
         dept\_name char(20),  
         building char(15),  
         budget numeric(12,2));  
         创建了表department和它的列，并指定了各列的数据类型。DDL语句会同时更新数据字典。
      4. 从应用程序访问数据库  
         SQL语句没有通用的图灵机那么强大，有很多计算是SQL无法做到的，比如接收用户输入、显示内容、网络通讯等，这些功能必须使用像c、c++等宿主(host)语言来实现，然后在程序中嵌入SQL语言来访问数据库。  
         从应用程序访问数据库有两种方式：
         1. 提供应用程序接口来传递DML、DDL，然后取回结果。比如c语言的ODBC、java的JDBC等；
         2. 在应用程序中使用DML预编译器，然后通过扩展宿主语言的语法来嵌入DML调用。
   6. 数据库设计  
      查询语句的编写和schema的设计
      1. 设计过程
         1. 需求分析阶段
         2. 概念设计阶段，选择数据模型，将需求转化为schema，这个阶段的重点是对数据和数据关系进行描述。这个阶段要解决使用哪些(what)字段和怎样(how)组织这些字段的问题，前者是商业问题，后者才是计算机科学问题。  
            关于组织字段的的方式通常有两种，一种是使用E-R模型，一种是使用算法自动生成表。  
            这个阶段设计的schema需要满足对数据增删改查的要求，
         3. 实现阶段，包括逻辑设计和物理设计两步，逻辑设计将高度抽象的概念模型转换为数据库实现，然后物理设计则解决文件的组织、存储结构等问题。
      2. 设计大学组织数据库
      3. E-R模型  
         entity相当于现实世界中的事物，它用字段(attribute)的集合来描述，Eneity之间的关联关系称为relationship。可以用E-R图来反映实体和关联关系，常用UML(Unified Modeling Language)来绘制：  
           
         Entity用矩形来表示，实体名称放在头部，下面放置实体字段；Relationship用棱形表示，内部填写Relationship的名称。
      4. 规范化  
         规范化也是一种设计关系数据库的方法，它的目标是生成一个关系模式集合，使信息的存储没有冗余，而且还能高效的检索。  
         不好的设计会造成信息的冗余以及缺乏表达某些信息的能力：  
           
         比如这张表同时包含了教师(id,name,salary)和部门信息，history信息在两行中都有，修改history的budget时需要两行都修改；而如果我们想增加一个部门时，就会暴露出这个设计表达能力不足的问题，因为只有先增加一个教师才能增加部门。
   7. 数据存储和检索  
      对一个数据库系统来说，存储管理器和查询处理器是非常重要的两部分。
      1. 存储管理器  
         数据存储在硬盘上，CPU的和内存的速度大于硬盘的读取速度，所以它需要尽可能地减少数据在硬盘与内存之间的转移，并负责将DML语句翻译成物理层的命令。即存储管理器负责数据的增删改查，它包含的模块或功能有：  
         权限及完整性管理器：负责检查完整性约束、控制用户访问权限；  
         事务管理器：用于保证系统在发生故障时的一致（正确）状态，以及保证并发事物无冲突地执行；  
         文件管理器：负责硬盘空间的分配并管理保存在硬盘上的数据结构；  
         缓存管理器：负责将数据从硬盘读取到内存，并决定将哪些数据缓存到内存。缓存管理器是数据库系统的重要部分，有了缓存管理器，数据库才能够处理比内存容积大得多的数据。  
         存储管理器所操作的文件类型有：  
         数据文件：数据库本身  
         数据字典：对schema的描述  
         索引：用于提高检索效率，索引维护了一系列指向特定数据的指针
      2. 查询处理器  
         包含多个部分：
         1. DDL翻译器：翻译DDL指令并记录到数据字典
         2. DML编译器：将DML语句编译成查询评估引擎能理解的低级指令。DML语句往往可以有不同的翻译，这些翻译的执行结果相同但性能不一，DML编译器也负责查询优化，选择性能最高的翻译方式。
         3. 查询评估引擎：执行由DML编译器生成的低级指令
   8. 事务管理  
      某些对数据的的操作常常会形成单一的逻辑工作单元，比如转账，从A账户扣款M元，然后给B账户增加M元，这两步操作必须要么都成功，要么都失败，这样的要求也称为原子性（atomicity）；另外A、B账户的总和在操作事务前后保持不变，这称为一致性（consistency）。事务操作必须满足原子性和一致性，在操作过程中允许暂时地违背一致性（比如转账，A、B账户的增减需要过程）。  
      一致性由开发人员来保证，而原子性由数据库自身来保证；  
      对于转账操作来说，开发人员需要合理地划分操作步骤，比如讲之分为从A账户转出和给B账户转入两步，如果将这两步划分为一个事务，则符合一致性，任何一部独立，则会违背一致性；  
      数据库的恢复管理器负(recovery manager)负责原子性，实际应用中，事务难免会失败，这时要屏蔽失败的事务对数据库的影响（回滚到失败前的状态failure recovery）。  
      并发控制器（concurrency-control manager），在遇到事务需要并发执行时，无法保存操作前的状态，这时由并发控制器通过控制事务间的交互来保证一致性。  
      恢复管理器和并发控制器组成了事务管理器。
   9. 数据库结构(database architecture)  
      数据库的结构多种多样，比如有集中式(centralize)、服务端-客户端式、并行式(parallel)、分布式(distrubuted)等。  
      数据库的通用结构如图：  
        
        
      cs与bs架构下的数据库系统
   10. 数据挖掘和信息检索  
       knowledges discovery in databases  
       s set of rules  
       information retrieval--from textual data
   11. 特殊的数据库
       1. Object-based data models
       2. Semistructured data models
   12. 数据库的普通用户和管理员  
       数据库的用户可以分为普通用户(user)和管理员(administrator)
       1. 普通用户及对应的用户接口  
          依据用户期望的与数据库的交互方式可以进一步分为四类，对应有不同的用户接口
          1. 无经验的用户(naive user)：使用事先写好的程序与数据库交互，用户接口通常为表单；
          2. 开发者：负责编写数据库交互界面；
          3. 熟练的用户(sophisticate)：比如数据分析员，使用专业的分析软件或直接写SQL来与数据库交互；
          4. 专门的用户(specialized user)，编写特殊的数据库应用的人员，比如计算机辅助设计系统、知识库、专家系统、存储复杂数据结构（声音、图像）的系统等。
       2. 管理员(DBA)  
          通常DBA可以做的操作包括：
          1. 定义schema
          2. 定义数据的存储结构和访问方式
          3. 修改schema和数据的物理组织方式
          4. 访问权限控制
          5. 日常维护，比如定期备份、确保硬盘空间充足、监控数据库高效运行等