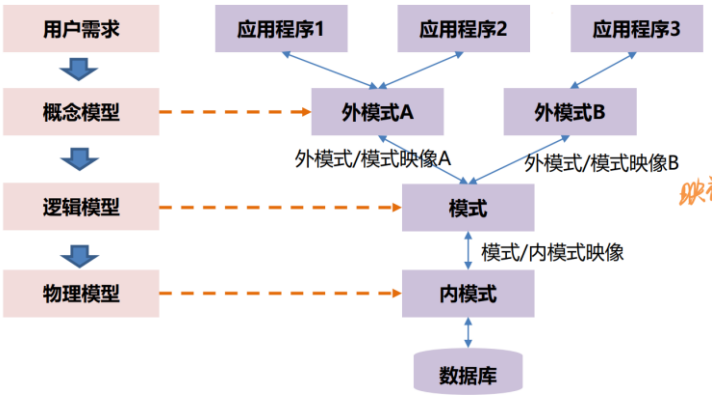
绿色**是复习课上老师亲口提到的**
红色**是我的强调**
蓝色**是历年题出现的**

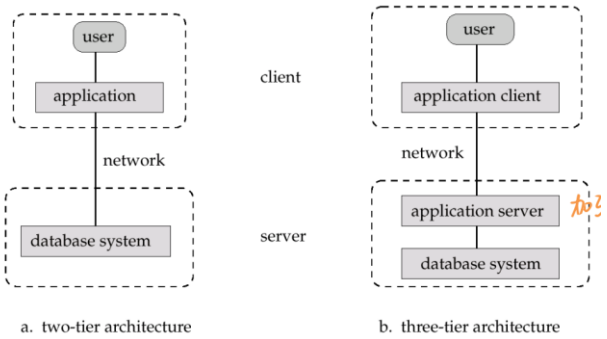
数据是数据库中存储的基本对象
数据的类型：结构化数据（关系数据（表））、半结构化数据、非结构化数据
数据库是一组相互有关联的数据集合，长期储存在计算机中，有组织、可管理和可共享
数据库中存储的是数据以及数据之间的联系
数据库的基本特征：数据按一定的数据模型组织、描述和储存；支持数据的增删改查；支持并发查询处理
数据库系统是指由数据库管理系统和相关工具组成的软件系统，用于管理和操作大量数据
一般包括 • 数据库 • 数据库管理系统 • 开发工具、应用系统 • 数据库管理员和终端用户
数据库管理系统是管理数据库的软件

传统文件处理系统的缺点 • 数据的冗余和不一致性 • 数据访问困难 • 数据孤立 • 完整性问题 • 原子性问题 • 并发访问异常 • 安全性问题
数据抽象的层次 • 物理层 • 逻辑层 • 视图层
模式(Schema): 数据库的逻辑结构 • 物理模式/内模式/存储模式 • 逻辑模式/模式 • 外模式/子模式（对应三种层次）
数据库的三级模式结构中，描述数据库中全体数据的全局逻辑结构和特征的是**逻辑模式**
数据库的二级映像为**外模式/模式映像**和**模式/内模式映像**
实例(Instance) 特定时刻存储在数据库中信息的集合称为数据库的一个实例
模式和实例之间的关系类似编程语言中**数据类型和变量**之间的关系
物理数据独立性(Physical data independence) 改变物理模式不改变逻辑模式
用户的应用程序与存储在磁盘上数据库中的数据是相互独立的
逻辑数据独立性(Logical data independence) 改变逻辑模式不改变外模式
用户的应用程序与数据库中数据的逻辑结构相互独立
数据模型：• 关系模型(Relational model) • 实体-联系模型(Entity-Relationship data model) • 半结构化数据模型(Semi-structured data model) • 基于对象的数据模型(Object-based data models) • 其他数据模型
数据模型是由**数据结构、数据操作和完整性约束**三部分组成的
数据模型和数据库模式之间的关系：



数据库语言
• 数据定义语言（Data-definition language, DDL)

• 数据操纵语言（Data-manipulation language, DML）– • 过程式 DML • 声明型 DML
数据库引擎 • 存储管理器(Storage manager) • 查询处理器(Query processor) • 事务管理器(Transaction manager)
数据库应用系统结构 • 两层体系结构 • 三层体系结构



a. two-tier architecture　　b. three-tier architecture

数据库管理员(Database Administrator, DBA) DBA 是数据库系统中所有活动的管理者和协调者 **主要职责**包括 • 安全和完整性控制 • 数据库性能监控、分析与优化 • 数据库恢复 • 数据库结构调整 **!!!! 没有事务处理与并发控制（来自于历年题而非课本，考了不止一次）**

六个基本关系代数操作包括∪，−，π，σ，×，ρ（选择、投影、重命名是一元运算，集合并、集合差、笛卡尔积是二元运算），其他操作都可以由这些操作推导得到。
笛卡尔积不去重复属性，自然连接去掉重复属性 **要记得!!!!!!!!**
投影和集合并掉去重行 **要记得!!!!!!!!**
branch 银行 customer 银行客户 depositor 存款人
account 存款 borrower 贷款人 loan 贷款
属性的值通常要求是原子的(atomic) 多值属性和复合属性不是原子的
(A1,A2)叫**关系模式**，r(A1,A2)叫**关系**
把所有数据存在一个关系里面会导致
• **数据冗余**（一个系有很多学生，每个数据都把系和学生存进去）
• **数据稀疏**（把学生和所有课程都存进去，但是一个学生只上几门课，剩下的都是 null）
超码、候选码、主码可以确定一个关系中的所有属性（确定即唯一标识一个元组）
参照完整性约束(Referential integrity constraint)：对于参照关系中的每个元组来说，它在外码属性上的取值肯定等于被参照关系中某个元组在主码上的取值
查询语言类别 • 过程式/函数式（关系代数） • 非过程式/声明式（SQL 结构化查询语言、元组关系演算、域关系演算）

DDL 包括：关系模式、属性的域(domain)、完整性约束、索引结构、安全性和权限信息、物理存储结构、视图、授权
DML 包括：查询、插入、删除、更新、事务处理
Select 不去重（select distinct） 集合运算并 union 交 intersect 差 except 都自动去重 （union all）
全局约束是 **check 子句和断言**
数据库的安全性 • 数据库系统层次：允许特定用户仅访问所需数据的身份验证和授权机制
• 操作系统层次：操作系统超级用户可以对数据库执行任何操作

• 网络层次：使用加密来防止：【窃听：未经授权阅读信息【冒充：伪装成授权用户或发送据称来自授权用户的信息

防止恶意窃取或修改数据 • 物理层次：传统的锁和密钥安全 护计算机免受洪水、火灾等的影响 • 人为层次：用户必须确保不向入侵者授予授权 用户接受密码选择和保密方面的培训

在**视图**上可以查询、定义新的视图、更新视图，不能定义新的关系

物化视图：不仅存查询语句，还存查询结果 **缺点**：当数据更新时，物化视图要跟着更新（延后），实时性不好

实体完整性约束(Integrity Constraints)

• 域约束 create domain Dollars numeric(12, 2) • 非空约束 not null • check 子句

• unique 约束 • 参照完整性(Referential Integrity) foreign key (account_number) references account • （主码也算吧，怎么没写

角色：在数据库中建立一个角色集，可以给角色授予权限，就和给每个用户授权的方式完全一样，每个数据库用户被授予一组他有权扮演的角色（可以为空）

数据库设计(Database Design)

• Conceptual design(概念设计) E-R 图

• Logical design(逻辑设计) 关系模型、规范化 3NF

• Physical design(物理设计) 数据表设计（数据类型、完整性约束）、构建索引结构

E-R 图三个基本概念：实体集、联系集和属性

联系集的度/阶：一般二元，也有三元

物理存储介质的分类

• 速度、成本、存储易失性 volatile/ non-volatile

• 基本存储 primary storage（易失）：高速缓冲存储器 cache 主存储器 main memory
辅助存储 secondary storage（非易失）：快闪存储器 flash 磁盘 magnetic disk
三级存储 tertiary storage（非易失）：光盘 optical storage 磁带 tape storage

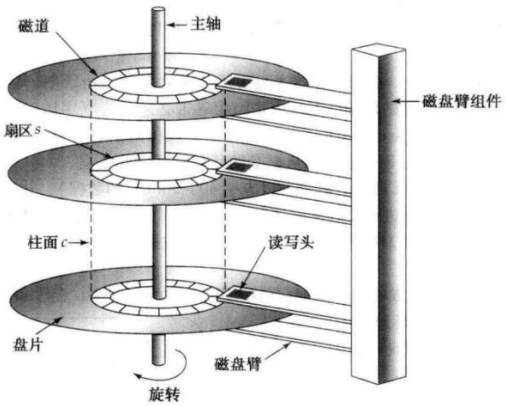


图 10-2 磁盘移动头机制

扇区是从磁盘读出和写入信息的最小单位

读写头将信息磁化存储到扇区中

磁盘性能度量指标：容量、访问时间、数据传输率、可靠性

一个**块**是一个逻辑单元，它包含固定数目的连续扇区

磁盘臂调度 电梯算法 从最内到最外，直到没有对更外的请求，开始调转向内侧移动

每个文件分成定长的存储单元，称为块

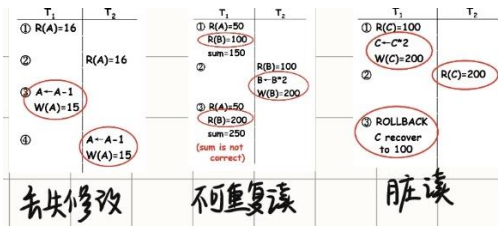
块是存储分配和数据传输的基本单元

搜索码是用于在文件中查找记录的属性或属性集
顺序索引是基于值的顺序排序
散列索引是基于将值平均分布到若干散列桶中
聚集索引（主索引）是文件按搜索码顺序排序
非聚集索引（辅助索引）是搜索码指定顺序和文件的记录物理顺序不同
使用辅助索引进行顺序扫描的效率不高，因为建立索引的搜索码的顺序与文件的存储顺序不一致，从而需要反复读取索引中的数据指针和存储的数据文件，其效率远低于在主索引上进行的顺序扫描。
稠密索引是每个搜索码值都有一个索引项
稀疏索引是只为搜索码的某些值建立索引项
顺序索引和哈希索引的区别（答案是老师亲口说的，不是书上的） • 顺序索引要维护有序性 哈希索引没有顺序是放进桶里 • 范围查询更适合用顺序索引 等值索引更适合用哈希索引

查询处理三个步骤: • 语法分析与翻译 Parser and translator • 优化 Optimization • 执行 Evaluation
度量查询计划的代价: 传送磁盘块数 t 和搜索磁盘次数 s
三种连接操作的区别: 嵌套循环连接是以元组为单位顺序扫描，块嵌套循环连接是以块为单位顺序扫描，索引嵌套循环连接是在内层循环时用索引查找代替文件扫描
物化是计算的每个中间结果被创建（物化），然后用于下一层的计算
流水线是通过将多个关系操作组合成一个操作的流水线来实现减少临时文件数，其中一个操作的结果被传送到下一个操作

事务 ACID 特性 • 原子性 atomicity • 一致性 consistency • 隔离性 isolation • 持久性 durability
事务的状态: • 活动的 • 部分提交的 • 失败的 • 中止的 • 提交的

并发控制这里所有都很重要!!!!!!!
并发控制中的问题：
• **丢失修改**: 两个事务 T1 和 T2 读入同一数据并修改，T2 提交的结果破坏了（覆盖了）T1 提交的结果，导致 T1 的修改被丢失。
• **不可重复读**: 不可重复读是指事务 T1 读取数据后，事务 T2 执行更新操作，使 T1 无法再现前一次读取结果。
• **脏读**: 读“脏”数据是指事务 T1 修改某一数据，并将其写回磁盘，事务 T2 读取同一数据后，T1 由于某种原因被撤销，这时 T1 已修改过的数据恢复原值，T2 读到的数据就与数据库中的数据不一致，则 T2 读到的数据就为“脏”数据，即不正确的数据。
举例:



两阶段锁协议 The Two-phase Locking Protocol：
增长阶段：事务可以获得锁，但不能释放锁
缩减阶段：事务可以释放锁，但不能获得锁
严格两阶段锁协议 Strict Two-phase Locking：除了要求封锁是两阶段以外，还要求事务持有的所有排他锁必须在事务提交后方可释放。这个要求保证未提交事务所写的任何数据在该事务提交之前均以排他方式加锁，防止其他事务读这些数据
强两阶段锁：要求事务提交之前不得释放任何锁，容易验证在强两阶段封锁条件下，事务可以按其提交的顺序可串行化
饥饿是一个事务可能永远不能取得进展的状态 **举例：**T2 在一数据项上持有共享锁，T1 申请对该数据项加排他锁，显然需要等待 T2 释放共享锁，同时，T3 可能申请对该数据项加共享锁，加锁请求与 T2 锁相容，因此可以授权 T3 加共享锁，此时 T2 可能释放锁，但 T1 还必须等待 T3 完成，又可能有 T4 在 T3 完成前申请加共享锁，事实上，可能存在一个事务序列，其中每个事务都申请对该数据项加共享锁，每个事务在授权加锁一小段时间后释放锁，而 T1 总是不能在该数据项上加排他锁，T1 可能永远不能取得进展，这称为饥饿。
死锁是事务集合中的每个事务在等待该集合中的另一个事务，没有一个事务能够取得进展的状态
举例：相互等待

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

基于锁的协议都**不能避免死锁**
可以避免死锁的是基于图的协议和基于时间戳的协议
死锁的预防：• 基于图的协议 The Graph-based Protocol • 基于时间戳的协议 The Timestamp-based Protocol（wait-die 机制基于非抢占技术/wound-wait 机制基于抢占技术）
• 锁超时（如果一个事务的等待时间超过了规定的期限，就认为发生了死锁）
死锁检测：等待图（等待图中出现了回路，就代表发生了死锁）
死锁恢复：三个需要采取的动作：选择牺牲者、回滚（彻底回滚、部分回滚）、饥饿

故障分类：事务故障、系统故障、磁盘故障
数据库修改：延迟修改、即刻修改

Alter table student modify name char(15) not null alter 可以是修改也可以是添加
Create table 时要注意顺序，外码的被参照方先建表
是否需要 distinct
Insert into students values()
Order by age asc/desc
Create view stu(a1,a2…) as (select …from…where…)
Grant select on student to user

Revoke select on student from user
Delete from account where…
Update account set balance=balance*1.2
Natural left outer join
Create index excel on student(age)