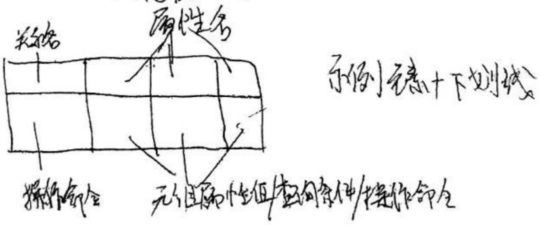
- DBS概述
  - 基本概念
    - 数据
    - 数据库
    - 数据库管理系统 ⇒ 定义，功能……
    - 数据库系统
  - 发展历程：人工管理 → 文件系统 → 数据库系统
    - 特点
      - ①
      - ②
      - ③
      - ④ …
- 数据模型 data model
  - 概念模型
    - 定义
    - 基本概念：①实体 ②属性 ③码 ④实体模型 ⑤实体集 ⑥联系
  - 转化
  - 表示方法：实体联系模型，层次(陈氏法)，鱼骨法……
  - 数据模型三要素
    - 数据结构
    - 数据操纵
    - 完整性描述约束
  - 层次模型，网状模型，关系模型
    - 要素
    - 优缺
    - 区别
  - 类
- 三级模式结构
  - schema 概念
  - 三级结构
    - 外模式
    - 模式
    - 内模式
  - 两级映像与数据独立性：外/模，模/内
- DBS组成
  - 硬件平台
  - 软件平台
  - 人员
    - DBA
    - 分析员、设计员
    - 应用程序员
    - 用户
- 体系结构
  - 集中式
  - 客户-服务器
  - 并行
  - 分布式
  - 云……

关系模型

- 基本数据结构及形式化定义
  - 关系（二维表）
    - 域：一组具有相同数据类型的值的集合，如 1, 3, 偶数集 ……
    - 笛卡尔积：符号 ×，n-tuple（n 元组），分量，基数（一个域允许的不同取值个数）
    - 关系：（单笛卡尔积无意义，其真子集才有意义）
      - 定义：$D_1 \times D_2 \times \cdots D_n$ 的子集 $R(D_1, D_2 \ldots D_n)$，n 是 degree
      - 类型：基本关系（实）、查询结果（临时）、视图（虚）
      - 基本关系性质：① 列同质（同一个域）② 不同列可出自同一个域 ③ 列、行顺序无所谓 ④ 任意两个元组 key 不能相同 ⑤ 分量必须取原子值
  - 关系模式（P1.32）
    - 定义：关系的描述称为关系模式，R(U, D, DOM, F) → 属性向域的映像集合，常说明为属性的类型、长度
    - 其他相关概念：候选码，主码（primary key），主属性，非主属性，全码
  - 关系数据库：支持关系模型的数据库系统
  - 存储结构：由 RDBMS 设计和实现
- 关系操作
  - 基本操作
    - 查询：select，project，join，divide，union，difference，intersection，笛卡尔积（5 种基本操作）
    - 更新：insert，delete，update
  - 特点：集合操作方式
  - 语言
    - 关系代数（ISBL）
    - 关系演算
      - 元组关系（ALPHA，QUEL）
      - 域关系（QBE）
    - SQL → 具有上二者双重特点
- 关系完整性
  - 实体完整性
  - 参照完整性
  - 用户定义的完整性
- 关系代数
  - 传统集合运算
    - 并 $R \cup S = \{t \mid t \in R \vee t \in S\}$
    - 差 $R - S = \{t \mid t \in R \wedge t \notin S\}$
    - 交 $R \cap S (= R - (R - S)) = \{t \mid t \in R \wedge t \in S\}$
    - 笛卡尔积 $R \times S = \{\widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S\}$
  - 专门的关系运算
    - 选择（限制）：$\sigma_F(R) = \{t \mid t \in R \wedge F(t) = \text{true}\}$ $F: X_1 \theta Y_1$（θ：比较运算符）
    - 投影：$\Pi_A(R) = \{t[A] \mid t \in R\}$ 如 $\Pi_{Sno,Smajor}(Student)$（A：R 中属性组列）
    - 连接（join），$R \underset{A\theta B}{\bowtie} S = \{\widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] \, \theta \, t_s[B]\}$
      - 等值连接
      - 自然连接（特殊的）
      - 外连接（左/右）
    - 除（division）：$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_Y(S) \subseteq Y_x\}$ → Y_x：x 在 R 中的象集，$x = t_r[X]$
- 关系演算
  - ALPHA
    - 检索
      - ① 简单 GET W(SC.Cno)
      - ② 带限定条件
      - ③ 带排序
      - ④ 指定返回元组条件
      - ⑤ 用元组变量的
      - ⑥ 用存在量词的
      - ⑦ 带多个关系表达式的
      - ⑧ 全称量词的 ⑨ 以 ↑ 3
      - ⑩ 用蕴涵的
      - ⑪ 集合函数
    - 更新
      - 修改：① HOLD ② 需主码别修改 UPDATE
      - 插入：PUT
      - 删除：① HOLD ② DELETE
  - QBE

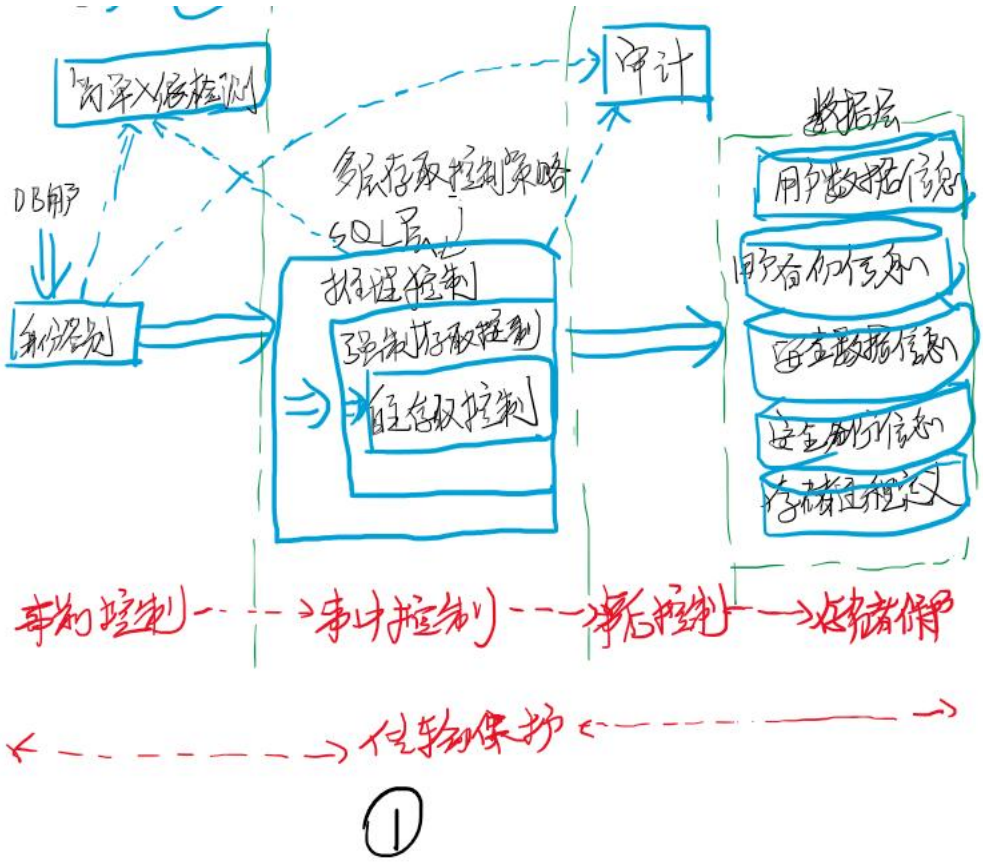


示例元素 + 下划线

SQL
- 概念：structured English query language
- 功能：data definition + data query + data manipulation + data control（支持分级授权）
- 特点/~~功能~~功能：综合且风格统一
  - ②数据操纵高度非过程化。[区别]：是否人工指定存取路径）
  - ③面向集合的操作方式。（对象为集合）
  - ④以统一语法结构提供多种使用方式。独立+嵌入
  - ⑤语言简洁且易学易用
- 运用
  - 数据定义
    - 模式 1.定义：CREATE SCHEMA[< >]AUTHORIZATION<> 2.删除：DROP Sm<>
    - 基本表 1.定义 2.数据类型 3.模式与表 4.修改 5.删除 RESTRICT / CASCADE
    - 索引 1.建立 2.修改 3.删除
    - 数据字典 记录所有定义信息
  - 数据查询
    - 单表 1.选择若干列/元组（DB中） 3.ORDER BY子句 4.聚集函数 5.GROUP BY子句 6.LIMIT子句
    - 连接 1.等值与非等值 2.自然连接 3.复合条件连接 4.自身连接 5.外连接 6.多表连接
    - 嵌套 1.带有 IN 谓词的子查询 2.带比较运算符的 3.带 ANY(SOME)或ALL的 4.带EXISTS
    - 集合 并UNION、交INTERSECT、差EXCEPT（结果列数、data type必须相同）
    - 派生表 derived table 成为主查询对象
  - 数据更新
    - 插入data：INSERT <表>[(<列>[,<列>]...)] VALUES(<常>[,<常>]...); （insert一个元组或子查询结果）
    - ~~删除~~修改：单元组 UPDATE <表> SET <列>=<式>[,<列>=<式>]... [WHERE <条>]；多个元组；嵌套查询（用以批量修改条件）
    - 删除：DELETE FROM <表> [WHERE <条>]；单/多个/嵌套查询
  - 空值处理
    - 定义 NULL ~~未~~①未知 ②不应有值 ③必须填写
    - 产生
    - 判断
    - 约束
    - 运算 见书P113表
  - 视图 view
    - 定义 1.建立 CREATE VIEW ... AS ... [WITH CHECK OPTION]；2.删除 DROP VIEW <表> [CASCADE]
    - 查询 遵先~~找~~到定义再进行对视图的查询
    - 更新
    - 作用（定义在基本表之上）：①保护机密 ③简化用户操作 ②对重构DB提供一定逻辑独立性 ④使用户对同一数据有不同看法

数据库安全性

- **概述**
  - 定义：保护数据库以防止不合法使用所造成的数据泄露、更改或破坏
  - 不安全因素
    - 非授权用户恶意存取、破坏
    - DB 重要/敏感数据被泄露
    - 安全环境的脆弱性
  - 安全性
    - 技术
    - 管理
    - 政策法律
    - 指南、评估、和解释者
  - 安全标准
    - TCSEC/TDI ⇒
      - A1 验证设计
      - B3 安全域
      - B2 结构化保护
      - B1 标记安全保护
      - C2 受控存取保护
      - C1 自主安全保护
      - D 最小保护
    - 指标
      - ① 安全策略
      - ② 责任
      - ③ 保证
      - ④ 文档
    - CC：EAL1~EAL7
- **控制**
  - 模型：见图简单中
  - 技术
    - 用户身份鉴别
    - 存取控制
    - 自主存取控制方法
    - 授权与权限回收
    - 数据库角色
    - 强制存取控制方法
- **视图机制**
  - 通过视图机制把要保密的数据对无权存取这些数据的用户隐藏起来，从而提供安全保护
  - 配合授权机制，间接实现支持存取谓词的用户权限定义
- **审计**（C2以上）
  - 审计事件
    - 服务器事件
    - 系统权限
    - 语句事件
    - 模式对象事件
  - 审计功能
    - 基本功能
    - 多套审计规则
    - 提供审计分析的报表功能
    - 审计日志管理功能
  - AUDIT（设置）/NOAUDIT（取消）
    - 用户级
    - 系统级
- **数据加密**
  - 数据加密
  - 加密基本思想
  - 方法
    - 存储加密
      - 透明
      - 非透明
    - 传输加密
      - 链路
      - 端到端
- **其他安全性保护**
  - 推理控制
  - 隐蔽信道
  - 数据隐私
  - “三权分立”安全管理机制


审计
DB用户
多层存取控制策略
SQL层
数据层
用户数据信息
用户备份信息
安全数据信息
安全备份信息
存储过程定义
事前控制
事中控制
事后控制
恢复备份
传输加保护

①

数据库完整性

概述

定义：指数据库数据的正确性和相容性

必备功能：
- 提供定义约束的机制
- 提供检查约束的方法
- 提供违约处理方法

实体完整性

定义方法：CREATE TABLE + PRIMARY KEY

说明：
- 列级约束：单一属性构成的码（直接跟在属性后）
- 表级约束：单、多属性组成的码（所有属性后边，加括号）

检查和违约处理：每次插入/更新 → ①检查主码值是否唯一，否则拒绝插入/修改；②检查主码各属性是否为空，有空则拒绝

方法：全表扫描（低时），B+树索引

参照完整性

定义方法：FOREIGN KEY 定义哪些列为外码 REFERENCES 引用哪个表的哪个码

检查和违约处理：

被参照表（Student）	参照表（SC）	违约处理
可能破坏参照完整性	← 插入元组	拒绝
可能破坏参照完整性	← 修改外码值	拒绝
删除元组	→ 可能破坏完整性	拒绝 / 级联删除 / 设为NULL
修改主码值	→ 可能破坏完整性	同上

用户定义的完整性

属性

定义：在CREAT TABLE中
1. 列值非空 NOT NULL
2. UNIQUE
3. CHECK 短语

（在属性值后）
- 不允许取NULL
- 取值唯一
- CHECK短语指定约束条件

属性上约束的检查和违约处理

区别

元组

约束定义：CHECK短语（可设置不同属性之间取值的相互约束）

检查和处理

命名子句

格式：CONSTRAINT <完整性约束名> <完整性约束>

NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK、etc.

修改表中的完整性限制：ALTER TABLE

域完整性

声明性断言...

触发器

定义：
1. 创建
2. trigger名
3. table名
4. trigger事件
5. trigger类型
6. 触发条件
7. 触发动作体

执行trigger：BEFORE → SQL → AFTER

删除触发器：DROP TRIGGER <trigger名> ON <table>

关系数据理论

- 问题的提出
  - 关系模式 $R(U, D, DOM, F)$ ——→ 模式设计 $R(U, F) \ni r$
  - 数据依赖
    - 函数依赖 FD
    - 多值依赖 MVD
    - → 含义：一个关系属性与属性间的约束关系，体现值相等与否，数据内在性质语义；抽象现实属性相互联系
  - 设计存在的问题
    - ① 数据冗余
    - ② 更新异常
    - ③ 插入异常
    - ④ 删除异常
    - 原因 → 依赖存在不好的性质 —解决→ 规范化

- 规范化
  - 函数依赖：给定 $R(U, F)$，X、Y 为 U 子集，对 $\forall r \in R$，若在 X 上属性相等，则 Y 上属性一定相等，
    - X 确定 Y（Y 依赖于 X）记为 $X \rightarrow Y$
    - 非平凡的函数依赖：$X \rightarrow Y$ 且 $Y \not\subseteq X$
    - 平凡的：$X \rightarrow Y$ 且 $Y \subseteq X$
    - 决定因素：$X \rightarrow Y$，则 X 为决定因素
    - $X \leftarrow\rightarrow Y$，当 $X \rightarrow Y$ 且 $Y \rightarrow X$
    - $X \not\rightarrow Y$，Y 不依赖于 X
    - 完全函数依赖：$R(U, F)$ 中，若 $X \rightarrow Y$ 且 X 任一真子集 X'，都有 $X' \not\rightarrow Y$，则称 $X \xrightarrow{F} Y$
      - $X \rightarrow F$ 但不完全依赖，则称为 $X \xrightarrow{P} Y$ 部分函数依赖
    - 传递函数依赖：$R(U, F)$ 中，若 $X \rightarrow Y (Y \not\subseteq X)$，$Y \not\rightarrow X$，$Y \rightarrow Z$，$Z \not\subseteq Y$，则称 Z 对 X 传递函数依赖，记作 $X \xrightarrow{T} Z$
  - 码：
    - 用函数依赖的概念定义：设 K 为 $R(U, F)$ 中的属性(组)，若 $K \xrightarrow{F} U$，则 K 为 R 候选码
    - 若 U 部分函数依赖于 K，即 $K \xrightarrow{P} U$，则 K 称为超码（候选码的超集）
  - 范式
    - $4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$
    - 1NF 作为二维表，关系要符合一个最基本的条件：每个分量必须是不可分的数据项。满足了这个条件的关系模式就属于第一范式（1NF）
    - 2NF 若 $R \in 1NF$ 且每一个非主属性完全函数依赖于任何一个候选码，则 $R \in 2NF$
      - 若 $R \notin 2NF$ 会产生以下问题
        - ① 插入异常
        - ② 删除异常
        - ③ 修改异常、
    - 3NF 设 $R(U, F) \in 1NF$，若 R 中不存在这样的码 X、属性组 Y 及非主属性 $Z (Y \not\supseteq Z)$，使 $X \rightarrow Y$，$Y \rightarrow Z$ 成立，$Y \not\rightarrow X$，则称 $R(U, F) \in 3NF$
    - BCNF 定义：$R(U, F) \in 1NF$，若 $X \rightarrow Y$ 且 $Y \not\subseteq X$ 时 X 必含有码，则 $R(U, F) \in BCNF$
      - 即 R 中每一个决定因素都包含码。满足 BCNF 的 R 性质 ①②③
    - 4NF 多值依赖才考虑，定义：$R(U, F) \in 1NF$，若对 R 的每个非平凡多值依赖 $X \rightarrow\rightarrow Y (Y \not\subseteq X)$，X 都含有码，则称 $R(U, F) \in 4NF$

- 数据依赖的公理系统
  - 对 $R(U, F)$，其任一关系 r，若 $X \rightarrow Y$ 都成立，则称 F 逻辑蕴涵 $X \rightarrow Y$，F 为函数依赖集
  - Armstrong 推理规则
    - A1 自反律：$Y \subseteq X \subseteq U$，则 $X \rightarrow Y$ 为 F 所蕴涵
    - A2 增广律：若 $X \rightarrow Y$ 为 F 所蕴涵，且 $Z \subseteq U$，则 $XZ \rightarrow YZ$ 为 F 所蕴涵
    - A3 传递律：若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴涵，则 $X \rightarrow Z$ 为 F 所蕴涵
    - A…公理系统 { 有效性、完备性
    - 推论
      - 合并规则：由 $X \rightarrow Y$，$X \rightarrow Z$ 有 $X \rightarrow YZ$
      - 伪传递规则：由 $X \rightarrow Y$ 及 ~~Z⊆Y 有 X→Z~~ $WY \rightarrow Z$，有 $XW \rightarrow Z$
      - 分解规则：由 $X \rightarrow Y$ 及 $Z \subseteq Y$，有 $X \rightarrow Z$，
      - → 引理：$X \rightarrow A_1 A_2 \cdots A_k \Leftrightarrow X \rightarrow A_i$ 成立 $(i = 1, \cdots, k)$
  - 闭包
    - 在 R 中为 F 所逻辑蕴涵的函数依赖的全体叫作 F 的闭包，记为 F^+
    - 若 $G^+ = F^+$，就说 F 覆盖 G
  - 每个函数依赖集 F 均等价于一个极小函数依赖集 F_m

- 保持函数依赖的模式分解
  - R 的一个分解 $\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \cdots, R_n(U_n, F_n)\}$
  - 其中 $U = \bigcup_{i=1}^{n} U_i$，且不存在 $U_i \subseteq U_j$，$1 \le i, j \le n$，F_i 是 F 在 U_i 上的投影
  - 即 $F_i = \{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq U_i\}$

数据库设计

概述
- 设计定义：构建优化的数据库逻辑和物理结构，满足用户需求
- 特点：重视基础数据，设计与处理相结合
- 方法：包括多个经验法和多种规范设计方法
- 步骤：6个阶段：需求分析 → 概念设计 → 逻辑设计 → 物理设计 → 数据库实施 → 运行维护
- 各级模式：需求、概念、逻辑、物理设计阶段分别形成不同模式

需求分析
- 分析任务：调查对象，了解原系统，确定新系统功能和用户需求
- 分析方法：通过多种调查步骤和方法，用SA方法分析表达需求
- 数据字典：包含数据项、结构、流、存储和处理过程

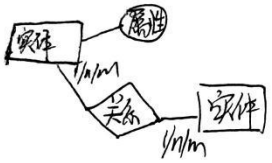
概念结构设计
- 概念模型：特点、①是真实反映现实 ②易理解 ③易更改和转换
- E-R模型：实体联系包括1:1、1:n、m:n，用图形表示实体属性和联系
- 扩展E-R模型：含ISA、基数约束、Part-of、弱实体等概念
- UML表示：类图中类对应实体，关联表示联系，还有基数约束等
- 设计过程：遵循实体属性划分原则，通过合并和重构集成E-R图

逻辑结构设计
- 主要原则：实体型和联系按规则转为关系模式，相同码可合并
- 模型优化：确定数据依赖、消除冗余、按范式和需求优化
- 外模式设计：使用别名，针对不同用户定义视图，简化复杂查询

物理结构设计
- 设计内容：确定存取方法和存储结构，考虑事务和DBMS特征
- 存取方法：B+树、哈希、聚簇索引方法的选择及适用条件
- 存储结构：确定数据、存放位置和系统配置，权衡多方面因素
- 结构评价：存储空间、存取时间、维护代价

数据库实施与维护
- 数据载入：抽取、转换、输入数据，利用工具提高效率
- 程序调试：与数据库设计同步，按软件~~工程~~方法进行
- 数据库试运行：包括功能和性能测试，根据结果调整设计
- 运行维护：DBA负责转储恢复、安全控制、性能监督和重组织等工作

E-R模型：



关系查询处理与优化

- RDBMS的查询处理
  - 四步骤：
    - ① 查询分析：扫描查询语句，分析词法语法（语法正确性合法性）
    - ② 查询检查：根据语义，用语法树生成
    - ③ 查询优化：众多策略中选一个高效执行的计划（Explain查看查询计划）
    - ④ 查询执行：优化器执行策略 —生成→ 查询执行计划，生成代码执行→结果
  - 实现查询操作的算法
    - 全表扫描：对小表简单有效，对大表选择率低时效率很低
    - 索引扫描：选择率低优于FullTableScan；选择率高或元组均匀分布，则低效
    - ⋈：
      - 嵌套循环算法(nested loop join)
      - 排序-合并算法(sort-merge join)
      - 索引连接算法(index join)
      - 哈希连接算法(hash join)
        - 划分阶段(create)
        - 试探阶段(join)
- RDBMS的优化
  - 查询优化(必要性)
    - 概述
      - 关系系统
      - 非关系系统
    - 优点
      - 为用户节省精力
      - 系统优化比用户更优(优化器)
    - 计算：代价模型 —计算→ 代价最小
      - 集中式DB：总代价=I/O+CPU+内存..
      - 分布式DB：总=I/O+CPU+内存+通信
      - 开销单位(块数)
    - 总目标：选择有效策略，尽量降低查询代价
  - 实例
- 代数优化(规则)
  - 语法树：是关系代数表达式的内部表示
  - 关系代数表达式等价变换规则 $E_1 \equiv E_2$，常见变换规则见教材
  - 典型启发式优化规则
    - ① 选择运算尽可能先做
    - ② 投影、选择同时进行
    - ③ 把投影同其前后的双目运算结合起来
    - ④ 把某些选择同在它前面要执行的笛卡儿积结合成连接运算
    - ⑤ 找出公共子表达式
- 物理优化
  - 代数：改变操作次序、组合，不涉及底层存取路径
  - 物理：选择高效合理的操作算法或存取路径
  - 优化方法
    - 基于启发式规则的启发式优化（大多时适用）
      - 选择
        - 小关系
        - 大关系
      - 连接
        - 是否排序
        - 有无索引
        - 大小表
    - 基于代价估算的优化：精细复杂
    - 两者结合的优化方法
- 查询计划的执行
  - 方式
    - 自顶向下：被动、需求驱动
    - 自底向上：主动
  - 生成：由RDBMS在查询优化完成之后

数据库恢复技术

- 事务的基本概念
  - 定义：是用户定义的一个数据库操作序列，这些操作要么全做，要么全不做，是一个不可分割的工作单位。（SQL可被嵌入程序）
    - → 一般为 BEGIN TRANSACTION; / COMMIT; / ROLLBACK;
  - 特性：ACID：
    - Atomicity 原子性
    - Consistency 一致性
    - Isolation 隔离性
    - Duration 持续性
- 数据库恢复：DBMS必须具有把DB从错误状态恢复到某一已知的正确状态的功能。
- 故障的种类：
  - 事务内部：事务故障无法预期的终点，要撤销事务（UNDO）
  - 系统故障（软）：指造成系统停止运转的任何事件，使得系统要重新启动
    - 系统重启后除UNDO外还要重做（REDO）所有已提交的事务，以使数据库恢复到一致状态
  - 介质故障（硬）：外存故障，可能性小破坏性大 → 对策：数据备份、冗余
- 恢复的实现技术
  - 数据转储：定期备份数据（后备副本），转储后备份后，还要重装运行事务，耗时耗资源
    - 静态转储：在下次转储开始运行事务时进行（即期间不允许存取、修改），保证一致性
    - 动态转储：转储期间允许对数据存取
    - 海量转储：每次转储全部数据库
    - 增量转储：每次只转储上一次转储后更新过的数据
  - 登记日志文件
    - 格式内容：①各事务开始标记 ②结束标记（COMMIT/ROLLBACK）③所有更新操作
      - 以记录为单位，日志内容：①事务标识 ②操作类型 ③操作对象 ④更新前旧值 ⑤更新后新值
    - 日志文件作用：①事务故障恢复和系统故障恢复 ②动态转储方式中必须建立以便恢复 ③静态转储为了方便 ~~并~~ 数据恢复也可写日志
    - 无力怎维护步骤：①周期性地执行建立检查点，保存数据库当前状态的操作
    - 恢复子系统的恢复策略：①从重新开始文件中找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录
      ②由该检查点记录得到建立前所有执行中的事务编号，UNDO-LIST和REDO-LIST（将ACTIVE-LIST暂时对放入UNDO-LIST队列，REDO队列暂为空
      ③从检查点开始正向扫描日志文件，对每个开始的事务，放入UNDO-LIST，即此前已提交的事务
      把Tj从UNDO-LIST → REDO-LIST直到日志结束
      ④对UNDO-LIST中每个事务UNDO，对REDO-LIST中每个事务REDO（起点可为T_c时刻）
  - 数据库镜像
    - 复制：应用1 update → 副本 → 应用2 read，应用3 read，应用4 read
    - 恢复：应用1 update → 恢复 → 副本 → 应用2 read，应用3 read，应用4 read
    - 缺陷：降低运行效率，往往只对关键性数据/日志文件镜像

并发控制

并发控制概述

为了保证事务的一致性和隔离性，DBMS需要对并发操作进行正确调度

并发导致的不一致性情况类型：
- ① 丢失修改：同一数据，T_1修改前，T_2也修改数据，T_1结果丢失
- ② 脱读：读取"数据"，T_1改动W(A)→T_2R(A)→T_1UNDO→"数据"不对
- ③ 不可重复读：T_1 R(x) → T_2 UPDATE(x)...W(x) → T_1R(x) → 前后不一致
- 幻读(phantom)：T_1R(x) → T_2 INSERT(x)/DELETE(x) → T_1R(x)失败

并发控制机制就是要用正确方式调度并发操作，方法有：封锁(locking)、时间戳(timestamp)、乐观控制法(Optimistic scheduler)、多版本并发控制(MVCC)

事务的隔离级别

级别	丢失修改	脱读	不可重复读	幻读
读未提交(RU)	不	可	可	可
读已提交(RC)	不	不	可	可
可重复读(RR)	不	不	不	可
可串行化	不	不	不	不

隔离级别不是越高越好，数据一致性增强的同时，系统代价也会升高

封锁

封锁：事务对某个数据对象施加控制，在释放前不能被其他T改变

基本类型：排他锁(写锁、X锁)：T对A加上X锁，只许T读、改A，直到T解放锁（其间A不能加其他锁）

共享型锁(读锁，S锁)：T对A加上S锁，则事务T可以读A不能改A，其他事务只能对A再加S锁，而不能加X锁，直到T释放A上S锁

相容矩阵：

T_1 \ T_2	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

封锁协议

一级封锁：T修改R之前先加X锁，T结束(COMMIT/ROLLBACK)后释放。防丢失修改，保证T可恢复

二级：在一级基础上，增加T在读取R之前必须先对其加S锁，读完释放

三级：一级基础上，增加T在读R前先对其加S锁，直到T结束释放

一致性保证：

协议	X锁		S锁		一致性保证			能否防止以下错误
	操作结束释放	T结束释放	操作结束释放	事务	不丢失	不脏读	可重复读	
1级		√			√			读未提交
2级		√	√		√	√		读已提交
3级		√		√	√	√	√	可串行化

活锁和死锁

livelock
- 现象：加锁的请求被拒绝后不断push，导致有些事务"饿到"的事务可能长时间排不到（是指stuck形成排队先后）
- 解决方法：先来先服务 stack→queue

deadlock
- 现象：T_1 R(lock) → R(new write) T_2 → T_1、T_2永远wait不能结束（T_1 holdlock, wait... R(lock)）
- 预防：一次封锁法；顺序封锁法
- 诊断与解除：超时法；事务等待图法

并发调度可串行性

可串行化调度：多个T的并发执行是正确的 ⟺ 其结果与按某次序串行执行的结果相同

冲突可串行化调度：可串行的充分条件（冲突操作：R(x)与W(x)，W(x)与W(x)）

通过交换T_1、T_2不冲突操作次序得到另一个调度S_c'，若S_c'串行，则S_c也串行

两段锁协议

所有事务必须分两个阶段对数据加锁和解锁：①获得封锁在对操作之前加锁；②释放一个锁后，事务不再申请获得其他锁

若并发所有T均遵守两段锁协议，则对它们所有S_c都是可串行化调度

也是充分条件