目录

第五章 规范化设计

5.1 关系模式的设计问题

关系模式的外延和内涵：

一个关系模式包括外延和内涵两方面内容：

1) 外延：关系、表或当前值，与时间有关（泛关系）
2) 内涵：对数据定义以及数据完整性约束的定义，包含关系、属性、域的定义和说明（泛关系模式）
   a) 静态约束：数据之间的联系（即数据依赖）、主键和值域的设计
   b) 动态约束：定义各种操作（插入删除修改）对关系值的影响

泛关系模式与数据库模式：有时需要将 $R(U)$ 分解为若干个表，前者是泛关系模式，后者是数据库模式

关系模式的冗余和异常：

1) 数据冗余：同一个数据在系统中多次重复出现
2) 操作异常：由于数据冗余造成的异常
   a) 修改异常：少修改导致不一致
   b) 插入异常：插入新值时，只能设置为空（插入课程但无人选课只能设空）
   c) 删除异常：删除某值，只能一起删除（删除课程把老师也删了）

5.2 函数依赖 FD

完整性约束：关系是域的笛卡尔积的子集，但不是所有子集都是有意义的

1) 依赖于值域的限制：由 DBMS 完整性子系统实现
2) 依赖于属性值之间的相等与否：数据依赖

函数依赖 FD：

1) 定义：设关系模式 $R(U)$ 中 X 和 Y 为属性集 U 的子集，对任意 $r \in R(U)$ 都有对任意两个元组 $t,s \in r$ 都有 $t[X]=s[X] \rightarrow t[Y]=s[Y]$，那么称 FD $X \rightarrow Y$ 在 $R(U)$ 上成立。
2) 语义：FD 表示了关系模式集 X 值域 Y 值的多对一联系。
3) 分类：
   a) 完全依赖 $X \overset{f}{\rightarrow} Y$：左部不可约依赖
   b) 部分依赖 $X \overset{p}{\rightarrow} Y$：左部可约
   c) 传递依赖 $X \overset{t}{\rightarrow} Y$：存在 Z 满足没有 $Y \not\subset Z$ 且没有 $Z \rightarrow X$，有 $X \rightarrow Z$ 和 $Z \rightarrow Y$
4) 函数依赖集的闭包：$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$ 其中 F 逻辑蕴含 X→Y

FD 推理规则：

1) 自反性：$ABC \to AB$，产生平凡的 FD
2) 增广性：$X \to Y$ 则 $XZ \to YZ$
3) 传递性：$X \to Y$ 且 $Y \to Z$ 则 $X \to Z$

FD 与关键码的关系：

1) **超键**：如果有 $X \xrightarrow{p} U$，X 是超键
2) **候选键**：如果有 $K \xrightarrow{f} U$，K 是候选键
3) **主属性**：候选键中的属性

求候选键的方法：

1) 不在 FD 右部出现的属性必在候选键中
2) 只在 FD 右部出现的属性必不在候选键中
3) 求属性集闭包
4) 如果闭包不包含全部属性，尝试添加一个属性继续第三步

属性集的闭包： $X_F^+ = \{A \mid A \in U, X \to A \in F^+\}$

求属性集闭包：不断求新的属性直至包含所有属性或不再扩大

例：设U＝{A,B,C,D,E,,G}，F={ AB→C，C→A，BC→D，ACD→B, D→EG, BE→C, CG→BD, CE→AG } 求 $(BD)_F^+$

解：设X=BD

① 令： X(0) =BD

② 计算 X(1) :=BD∪EG=BDEG ; (D→EG)

③ 计算 X(2):= BDEG ∪C=BCDEG ; (BE→C)

④ 计算 X(3):= BCDEG ∪A=ABCDEG;

(C→A, BC→D, CG→BD, CE→AG)

X(3)已包括所有属性，算法终止。$(BD)_F^+$ **=ABCDEG**

函数依赖集等价：闭包相等，证明方法：互为子集则 FD 集等价
证明某个 FD 冗余：去掉它证明函数依赖集等价

最小函数依赖集： F_{min}

判断方法：首先要是函数依赖集（即闭包相等）

1) F 右端无冗余属性
2) F 中没有冗余的函数依赖
3) F 左端没有冗余属性

求最小 FD 集：

例：已知F={ A→D，B→D，BD→CA，CD→B }，求 F_{min} 。

解：第一步：应用分解规则得：

F1={ A→D，B→D，BD→C，BD→A ，CD→B }

第二步：消去函数依赖左端的冗余属性(应用伪传递)

由B→D, BD→C, 可推出B→C, 所以BD→C的左端的D是多余的;

由B→D，BD→A, 可推出B→A, 所以BD→A的左端的D是多余的;

∴ F2={ A→D，B→D，B→C，B→A ，CD→B }

第三步：去除F 中冗余的FD (应用传递性)；

由 B→A，A→D，，可推出 B→D，所以 B→D 是多余的。

∴ F_{min}={ A→D，B→C，B→A ，CD→B }

5.3 关系模式的分解办法

模式分解的等价性问题：

1) 数据等价：模式分解后是否表示同样的数据
2) 依赖等价：模式分解后是否保持函数依赖

无损分解：（无损联接分解 p111）原关系在每个分解后模式下的投影级联起来等于原关系。

无损分解不一定保持依赖等价，但一定数据等价。（可能有异常）

举反例：R(ABC) A→B 分解为 AB BC

无损分解测试方法：

1) 法一：chase 过程（所有模式和属性建表）
   a) 每行一个模式，每列一个属性，构造 R ρ 表
   b) 出现的属性填 a_j，不出现的属性填 b_{ij}
   c) $X \to Y$，则 X 相同的值应该有 Y 相同的值，用 a 或最小 b 的替换
   d) 反复进行，如果有一行全为 a，则是无损分解
2) 法二：如果 R 只分解为两个关系模式 R1 和 R2，则是无损分解的充要条件为：$R1 \cap R2 \to R1 - R2$或$R2 - R1$

保持函数依赖的分解：保持 FD 的分解

函数依赖集的投影：$\pi_Z(F) = \{X \to Y \mid X \to Y \in F^+ \wedge X, Y \subseteq Z\}$

1) 注意是 F 闭包上的投影
2) 注意即使是两个元素也别漏了可能能相互决定

保持函数依赖的分解：每一个分解的投影的并逻辑蕴含 F：$\left[\bigcup_{i=1}^{k} \pi_{R_i}\left(F^{(可写+)}\right)\right]^+ = F^+$

5.4 关系模式的范式

关系模式的范式：规范化的关系模式

1) 第一范式 1NF：每个元素（列）不可再分
2) 第二范式 2NF：1NF 且所有非主属性完全依赖候选键（2NF A→B B→C 候选键 A）
3) 第三范式 3NF：1NF 且所有非主属性不传递依赖候选键
4) BCNF：分解不保持 FD
   a) 定义 1：所有属性不传递依赖候选键
   b) 定义 2：所有决定因素（即 FD 左边）是码（即包含某个候选键）
   c) 全码（候选键是所有属性）一定是 BCNF

判断方法：$BCNF \subset 3NF \subset 2NF \subset 1NF$

1) 求候选键，不要遗漏
2) 判断是否为 BCNF：每个 FD 左边是否为超键？
3) 判断是否为 3NF：非主属性对候选键传递依赖？
4) 判断是否为 2NF：非主属性对候选键部分依赖？
5) 判断是否为 1NF

5.5 关系模式的分解

分解为保持 FD 依赖的 3NF：不一定是无损分解

1) 求最小函数依赖集 F_{min}
2) 对不出现在左端和右端的属性单独建模式（写在 ρ 里）
3) （可省略）如果存在依赖 $X \to A$，且 $X \bigcup A = U$，则 ρ 即为 R
4) 对 Fmin 中每一个依赖 $X \to A_1 A_2 A_n$ 构成一个关系模式 $XA_1 A_2 A_n \in \rho$

改进：分解为保持 FD 依赖且无损连接性的 3NF：
如果分解不包含候选键，再随便加一个候选键模式

分解为无损连接的 BCNF：不一定保持 FD（也不一定可能保持）

1) 直接拿 F 做，初始值 $\rho = \{R\}$，求候选键
2) 判断 ρ 中模式是否已经是 BCNF，如果是分解结束
3) 否则挑出 ρ 中不是 BCNF 的模式 S 的一个决定因素不是码的 FD $X \to A$，用 $\{XA, S - A\}$ 代替 S
4) 重复 2, 3，最终形成一颗二叉树

模式设计方法的原则：

1) 特性：不一定都满足
   a) 分解后为 BCNF 或 3NF
   b) 无损分解
   c) 保持 FD 集

2) 原则：
   a) 表达性：数据等价（无损连接）和依赖等价（保持 FD）
   b) 分离性：属性间的间接联系用不同模式表达
   c) 最小冗余性：分解后模式个数和模式中属性应当最少

4NF：
1) 产生冗余、操作异常的原因：多值依赖 MVD（X→→Y，X 决定 Y 的一组值）
2) 多值依赖的互补性：X→→Y 则 X→→U-X-Y，函数依赖是多值依赖的特殊情况
3) 举例：课程有两个老师和三本指定参考书，但是连函数依赖都找不出（全码）

5NF（PJNF）：
1) 联接依赖是实现关系模式无损连接的一种语义约束

本章涉及方法：
1) 求候选键，主属性与非主属性
2) 求属性集闭包 X_F^+
3) 证明两个函数**依赖集等价** $F^+ = G^+$ 充要条件相互包含
4) 证明依赖集内某个 FD 冗余
5) 判断依赖集是否是最小函数依赖集 Fmin
6) **求最小函数依赖集 Fmin**
7) **判断分解数据等价（无损分解测试方法）**：chase 方法、交决定差方法
8) **判断分解依赖等价（保持 FD）**：投影的并
9) 给定模式判断 BCNF 3NF 2NF 1NF
10) 分解模式为保持 FD 的 3NF
11) 分解模式为保持 FD 且无损连接的 3NF
12) 分解模式为无损连接的 BCNF

第七章 数据库设计

7.1 数据库设计概述

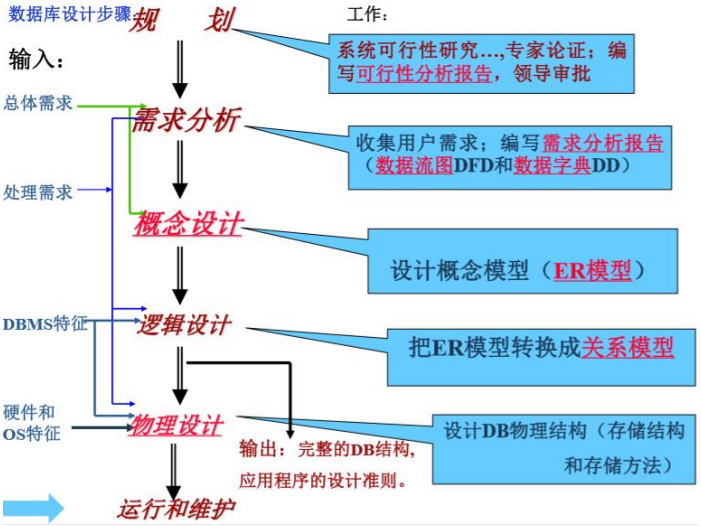
数据库设计 DBD：
对给定硬软件环境针对现实问题设计一个较优的数据模型，简历 DB 结构和 DB 应用系统
数据库设计对于一个给定的应用环境：

1) 一个确定最优数据模型与处理模式的逻辑块设计
2) 一个确定数据库存储结构与存取方法的物理设计
3) 建立起既能反映现实世界信息和信息联系，满足用户数据要求和加工要求，又能被某个数据库管理系统接受，同时能实现系统目标，并有效存取数据的数据库

软件生命周期：p141 六阶段
规划阶段、需求分析阶段、设计阶段、程序编制阶段、调试阶段、运行维护阶段

数据库系统生存期：p141 七阶段

1) **规划**：
   a) 建立数据库的必要性可行性分析，确定数据库系统在组织和信息系统中的地位，各数据库之间的关系
   b) 产生**可行性分析报告**和**数据库系统规划纲要**
2) **需求分析**：从数据库设计角度出发调查现实世界需要处理的对象，要确保用户目标一致性，获得需求
   a) 信息要求：用户将从数据库中获得的信息的内容和性质。最终导出数据库要求
   b) 处理要求：用户要完成什么功能，以及要求的响应方式和处理方式（批、联机）
   c) 安全性和完整性要求
   d) 产生**需求分析报告**（用户活动图、系统范围图、数据流图 DFD、数据字典 DD）
3) **概念设计**：将用户的信息要求等统一到一个逻辑结构中。概念结构要能表达用户需求，且独立于支持数据库的 DBMS 和硬件结构
   a) 步骤：抽象数据设计局部概念模式；将局部概念模式综合成全局概念模式；评审
   b) 方法：自顶向下、自底向上、逐步扩张（主要到次要）、混合策略
   c) 数据抽象的层次：聚集层次（姓名 is part of 技工）、概括层次（车 is a 交通工具）
   d) 产生 **ER 模型**
4) **逻辑设计**：形成逻辑数据库结构，将 ER 图转化为关系模型
   a) 数据库逻辑结构设计：设计与机器上 DBMS 所支持的数据模型相符合的逻辑结构，可以用 DDL 描述
   b) 应用程序设计：用主语言和 DBMS 的 DML 进行结构式程序设计
5) **物理设计**：得到完整、能实现的数据库结构
   a) 物理数据库结构的选择
   b) 逻辑设计中的程序模块说明的精确化
6) **实现**：将数据库逻辑设计和物理设计结果严格描述，调试产生目标模式
7) **运行与维护**：
   a) 收集和记录系统实际运行的数据，评价系统性能
   b) 保持数据库的完整性，有效处理数据故障和进行数据库恢复
   c) 对数据库进行修改或扩充



数据库设计的具体步骤：

1) 总体信息需求
2) 处理需求
3) DBMS 的特征
4) 硬件和 OS 特征

7.2 规划

规划阶段任务：

1) 确定系统范围
2) 确定开发工作所需要的资源
3) 估计软件开发成本
4) 确定项目进度

规划完成后，写出详尽的**可行性分析报告**和**数据库系统规划纲要**
将资料送交领导进行**审查和评价**

7.3 需求分析

对系统整个应用情况作全面的、详细的**调查**，确定企业组织的目标，收集系统总的设计目标的基础数据和对这些数据的要求，**确定用户的需求**，并把这些要求写成用户和数据库设计者能够接受的**文档**。

需求：信息需求、处理需求、安全性和完整性约束

输出**需求说明书：**系统的**数据流图**和**数据字典**。

需求分析的步骤：

1) 分析用户活动，产生**用户活动图**
2) 确定系统范围，产生**系统范围图**

3) 分析用户活动所涉及的数据，产生**数据流图 DFD**
4) 分析系统数据，产生**数据字典**
    a) 数据项
    b) 数据结构
    c) 数据流

7.4 概念设计

目标：产生概念模式（反映企业组织信息需求的数据库概念结构）
概念模式独立于计算机硬件结构，独立于支持数据库的 DBMS

概念设计的必要性：
1) 各阶段任务单一化，复杂度降低便于管理
2) 不受特定 DBMS 限制，独立于存储和效率的考虑，比逻辑模式更加稳定
3) 不含具体 DBMS 附加的技术细节，用户可以理解，反映用户需求

概念设计的步骤：
1) 进行数据抽象，设计局部概念模式
2) 将局部概念模式综合成全局概念模式
3) 评审

数据抽象：
数据库对象的形式：聚集（is part of）和概括（is a）
数据抽象的层次：概括层次、聚集层次

ER 模型的操作：
1) 实体类型的分裂
    a) 垂直分割：将实体的属性分割成若干组，记得保持主键出现在所有组中（教师变动信息、教师不变信息）
    b) 水平分割：把实体类型分裂为不相交的子类（男教师、女教师）
2) 实体类型的合并
3) 联系类型的分裂：担任关系分裂成主讲和辅导关系
4) 练习类型的合并

采用 ER 方法的数据库概念设计：
1) 设计局部 ER 模式
2) 设计全局 ER 模式
3) 全局 ER 模式的优化

7.5 逻辑设计

目标：将概念设计阶段的全局 ER 模式转变为与的 DBMS 所支持的数据模型相符合的数据结构（包括数据库外模式和模式）。模式在功能、完整性和一致性约束以及数据库的可扩充性方面应满足用户的各种要求。

逻辑设计的步骤：

1) 初始模式的形成
2) 子模式设计
3) 应用程序设计梗概
4) 模式评价
5) 修正模式

关系数据库的逻辑设计：

1) 导出初始关系模式
2) 规范化处理
3) 模式评价
4) 模式修正

7.6 物理设计

对于给定的基本数据模型选取一个最合适的应用环境的物理结构的过程。

1) 存储记录结构设计
2) 确定数据存放位置：记录聚簇
3) 存放方法设计
4) 完整性和安全性考虑
5) 程序设计

7.7 数据库的实现

根据逻辑设计和物理设计的结果，在计算机系统上建立起实际数据库结构、装入数据、测试和试运行的过程。

1) 建立实际数据库结构
2) 装入试验数据对应用程序进行调试
3) 装入实际数据进入试运行状态

7.8 数据库的运行和维护

1) 维护数据库的安全性与完整性
2) 监测并改善数据库运行性能
3) 根据用户要求对数据库现有功能进行扩充
4) 及时改正运行中发现的系统错误

第八章 数据库管理

8.1 事务的概念

事务：构成单一逻辑工作单元的操作集合
DBS 保证事务能正确、完整的执行

事务开始 BEGIN TRANSACTION：表示一个新事物开始
事务提交 COMMIT：数据库进入一个新的正确状态，事务对数据库的所有更新已经完成
事务回滚 ROLLBACK：事务执行不成功地结束，需要对所有更新撤销，恢复该事物到初始状态

事务的 ACID 性质：

1) 原子性 Atomicity：
   a) 一个事务对数据库的所有操作不可分割，要么全部执行要么全部不做
   b) 数据库系统本身的职责；由 DBMS 的事务管理子系统实现
2) 一致性 Consistency：（数据正确性）
   a) 事务独立之星的结果，应保持数据库的一致性，即数据不会因事务的执行遭受破坏
   b) 应用程序员的职责；DBMS 的完整性子系统执行测试任务
3) 隔离性 Isolation：
   a) 多个事务并发执行时，系统应当保证结果与事务先后单独执行的结果一致。即事务并发执行时保证执行结果正确
   b) 由 DBMS 的并发控制子系统实现
4) 持久性 Durability：
   a) 事务一旦全部完成后，对数据库的所有更新永久地反应在数据库中不会丢失，即使发生故障也是如此
   b) 由 DBMS 恢复管理子系统实现

8.2 数据库的恢复

数据库的可恢复性：系统能把数据库从破坏、不正确的状态恢复到最近一个正确的状态
实现方法：转储和建立日志文件
转储分类：静态/动态；全备份/增量备份

运行记录优先原则：为了安全，运行记录优先于操作

1) 写入日志后才写数据库
2) 所有日志写入完成后才能 commit

故障的分类：

1) **事务故障：**
   a) 故障只与某一个事务有关
   b) 分类：
      i. 可以预期的事务故障：存款金额透支，库存不够等，直接加 rollback 执行 undo
      ii. 非预期的事务故障：运算溢出，直接由系统执行 undo

2) **系统故障**：软故障
   a) 硬件故障、软件故障、掉电等引起系统停止运转重启的时间
   b) 特点：正在运行的所有事物都受影响，内存数据丢失，但数据库没有被破坏
   c) 对未完成事物进行 undo，再对已经提交的事物进行 redo
3) **介质故障**：硬故障
   a) 磁盘上的数据库遭到物理破坏
   b) 恢复：
      i. 从转储文件中恢复到转储时的正常状态
      ii. 根据日志文件找出已提交事物
      iii. 对已提交事物进行 redo，恢复到故障前的正确状态

检查点机制：为提高效率，只在检查点时刻才真正把所有修改写到磁盘，并在日志文件写入一条检查点记录

检查点恢复方法：
1) 正向扫描日志，对检查点-故障点间 commit 的事务加入 redo 队列，对未 commit 事务加入 undo 队列
2) 反向扫描 undo 队列，对每一个事务执行逆操作 undo
3) 正向扫描 redo 队列，对每一个事务进行 redo

8.3 数据的并发控制

并发控制带来的三个问题：（p177 图）主要原因就是事务的隔离性被破坏
1) 丢失更新：更新丢失了
2) 读脏数据：读到了修改后 rollback 的数据
3) 不一致分析（不可重复读）：记录被修改，记录被消失或增加了（幻行）

DBMS 的并行控制机制必须提供手段保证调度是可串行化的调度

封锁机制：p180
1) 排它锁（X 锁）：禁止其他读写
2) 共享锁（S 锁）：在所有 S 锁解除之前不允许加 X 锁

封锁的相容矩阵
封锁粒度（从整个数据库至某个物理块，粒度越大并发度越小）

封锁协议：
1) PX 协议：只有 XFIND R 获得 R 的 X 锁后才能读写 R，否则一直等。未规定释放时间。
2) PXC 协议：PX 协议加上只有在 commit 后解锁
   a) 解决丢失更新
3) PS 协议：任何更新记录必须要先执行 SFIND R 获得 S 锁，如果要更新必须用 UPDX R 操作升级为 X 锁，否则等待
4) PSC 协议：PS 协议加上只有在 commit 后解锁
   a) 进一步解决不一致分析和读脏数据问题
   b) **内存修改后 UPD 前加 UPDX！！记得写 commit 释放锁**

1) 活锁：一个服务永远处于等待状态；可采用 FIFO 或优先级解决
2) 饿死：由于 S 锁太多，始终没机会上 X 锁
3) 死锁：相互等待；可以用事务依赖图解决 p183；撤销环上的图

两段锁协议：

1) 对任何一个数据进行读写操作之前，事务必须获得对该数据的封锁
2) 在释放一个封锁后，事务不再允许获得任何其他封锁
3) 特点：
   a) 两段锁必然是可串行化的
   b) 可能使死锁增多，因为锁不能及时解除

8.4 数据库的完整性

完整性：数据的正确性、有效性、相容性

数据库的完整性规则：域完整性规则、基本表约束、断言

完整性子系统功能：

1) 监督事务执行，测试是否违反完整性规则
2) 如果有违反的情况，采取恰当的操作（拒绝、报告、改正）

完整性规则的组成：

1) 什么时候进行检查（触发条件）
2) 检查什么错误（约束条件或谓词）
3) 怎么处理错误（else 子句）

SQL 中的完整性约束：

1) 域约束：

   Create domain color char(6) default '???'
   Constraint valid_colors
   Check(value in ('red', 'yellow', 'green', '???'));

   Create table a (color color,…)

2) 基本表约束：
   a) 候选键：unique(列名, …)；primary key(列名, …)
   b) 外键：foreign key(列名,…) references(参照表)(列名) on delete 参照动作 on update 参照动作
      i. No action：对依赖表无影响
      ii. Cascade：级联删除或更新
      iii. Restrict：除非依赖表没有此外键值，否则拒绝删除或更新
      iv. Set null：设空
      v. Set default：设为默认值
   c) 检查约束 check：
      i. 只作用在基本表内部，可能产生违反约束的情况发生
      ii. 尽可能在 check 子句中不涉及其他表；应使用外键或断言代替
3) 断言：

   Create assertion ASS1 check(not exists/20 >= all (select count(distinct(pno)) from spj group by sno));

Drop assertion ASS1

SQL3 触发器：（主动规则、事件-条件-动作 ECA 规则）

触发器：由系统自动执行的对数据库修改的语句

1) 事件：指对数据库的插入删除修改操作，触发器开始工作
2) 条件：判断测试条件是否成立，如果不成立还是什么都不做
3) 动作：如果测试条件满足，执行的相应动作（数据库操作）

Create trigger TRIG1
After update (of price) on spj
Referencing
Old as oldtuple（删除和更新用）
New as newtuple（插入和更新用）
When (oldtuple.price > newtuple.price)
(begin atomic)
Update pcice = oldtuple.price
Where sno = newtuple.sno and pno = newtuple.pno and jno = newtuple.jno
(语句 2)
(end;)
For each row;（元组级触发器）
Drop TRIG1;
Create trigger TRIG2
Instead of update of cno on sc
Referencing
Old_table as oldstuff （只有修改前的元组）
New_table as newstuff （只有修改后的元组）
When (50>=all (select conut(sno)) from (sc except oldstuff) union newstuff group by cno) （SC-old 并 new）
Begin atomic
Delete from sc where (sno,cno,grade) in oldstuff;
Insert into sc select * from newstuff
End;

触发时间：

1) Before：测试 when 条件是否满足→执行触发器动作→执行触发事件
2) After：执行触发事件→测试 when 条件是否满足→执行触发器动作
3) Instead of：测试 when 条件是否满足→执行触发器动作/执行触发事件

触发事件： update、delete、insert

元组级触发器对每一个更新的元组检查结果
语句级触发器不允许应用修改前后的元组，但可以应用修改前后的元组集

存储过程：在服务器上预编译好 SQL 语句，一次编译多次运行，减少开发工作量和网络流量 p291

8.5 数据库的安全性

安全性级别 p191：环境级、职员级、OS 级、网络级、DBS 级

权限：
1) 访问数据权限：读、插入、修改、删除
2) 修改数据库结构权限：索引、资源、修改、撤销

视图的安全性优点：数据安全性、逻辑数据独立性、操作简便性

P193
授权：
Grant select, insert(sno), delete, update, references（定义外键）, usage（使用已经定义的域）
On s to usr1, usr2 (with grant option（可以传递权限）)

回收：
Revoke select on s from usr1 {restrict（如果有传给别人则拒绝回收）|cascade}

数据加密
自然环境的安全性

第十章 对象关系数据库

10.1 对象联系图

新一代 DBS：
1) ORDBS：对象-关系数据库
2) OODBS：面向对象数据库

数据模型：
1) 平面关系模型：
   a) 属性都是基本数据类型
   b) 数据结构层次：关系→元组→属性
   c) 1NF 关系
2) 嵌套关系模型：
   a) 关系的属性值是基本数据类型或关系类型（表）
   b) 数据结构可以多次嵌套
3) 复合对象模型：
   a) 属性是基本数据类型、关系类型（表）、元组
   b) 数据结构可以多次嵌套
4) 引用类型（面向对象模型）：
   a) 复合对象模型基础上，添加引用（指针）方式，并引入继承性

对象联系图的成分 p219：
1) 椭圆：对象类型（相当于实体类型）
2) 小圆圈：基本数据类型
3) 椭圆之间的边：对象之间的“引用”
4) 单箭头→：属性是单值的
5) 双箭头→→：属性是多值的
6) 双线箭头=>：超类和子类的联系（子类指向超类，学生指向人）
7) 双向箭头←→：两个属性之间值互为逆联系

数据细化是一种“是”（is a）联系
数据的细化/泛化构成子类、超类

10.2 面向对象的类型系统

面向对象技术中的数据类型系统：
1) **基本类型**：SQL 的基本类型整数、浮点、字符字符串、布尔、枚举
2) **复合类型**：
   a) 行类型：（元组类型、结构类型、对象类型）
      i. 例如：1, Oct, 2007
   b) 数组类型：相同类型元素的有序集合（不允许重复），长度预置

c) 列表类型：相同类型元素的有序集合（允许重复） select order by
d) 包类型：相同类型的无序集合（允许重复） select
e) 集合类型：相同类型的无序集合（不允许重复） select distinct

3) **引用类型**

聚集类型：数组、列表、包、集合类型

10.3 ORDB 定义语言

数据类型定义：

Create type date (day integer, month char(10), year integer)
Create type Mystring char varing
Create type courselist setof(Mystring)
Create type stucourse (name Mystring, cg setoff(coursegrade))

Create table sc of type stucourse
Create table sc (name Mystring, cg setoff(course Mystring, grade integer))

继承性：

类型级继承性：Create type student (degree Mystring, depth Mystring) under Person
表级继承性：create table student (degree Mystring, depth Mystring) under People

```
Create table university (
    Uname Mystring, city Mystring,
    President ref(faculty),
    Staff setof(ref(faculty)),
    Edit setof(ref(coursetext))
);
```

10.4 ORDB 查询语言

ORDB 查询语言对 select 的规定：

1) 允许 select 出现在任何关系名的任何地方
2) 所有 from 后面的表要有 as
3) 单值属性后面可以跟.，多值属性不行
4) 路径上包含集合时就不能继续写下去了