1 绪论

1.1 概述

数据是描述现实事物的符号记录
数据管理是数据处理的中心问题
三个发展阶段:

1. 人工管理
    1. 数据不长期保存
    2. 应用程序管理数据
    3. 数据不共享
    4. 数据不具有独立性
2. 文件系统
    1. 数据可长期保存
    2. 文件系统管理数据
    3. 数据共享性差、冗余度大
    4. 数据独立性差
3. 数据库系统
    1. 数据结构化
    2. DBMS 同一管理和控制数据
    3. 数据共享性高、冗余度低
    4. 数据独立性高

主要概念:

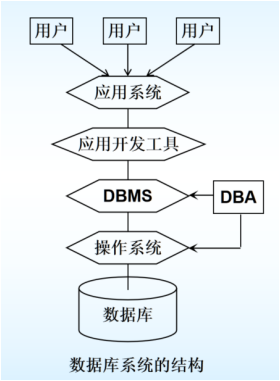

图 1: 数据库系统的结构

1. 数据库（DataBa 简称 DB）长期存储在计算机中内、有组织。可共享的数据集合

2. 数据库管理系统（DataBase Management System 简称 DBMS）专门用于管理数据库的软件

   主要功能：

   数据定义功能：由 DBMS 提供的数据定义语言 (Data Definition Language, DDL) 定义数据库中的数据对象。数据组织、存储和管理：分类组织、存储和管理各种数据，包括数据字典、用户数据、数据的存取路径等。

   数据操纵功能：由 DBMS 提供的数据操纵语言 (Data Manipulation Language, DML) 实现对数据库的查询、插入、删除和修改。

   数据控制功能：由 DBMS 提供的数据控制语言 (Data Control Language, DCL) 实现数据保护和事务管理等功能。包括完整性、安全性、并发控制、数据库恢复。**数据库的建立和维护功能**。

   其他功能：DBMS 与其他软件的通信，与另一个 DBMS 或文件系统进行数据交换；异构数据库的互操作等。

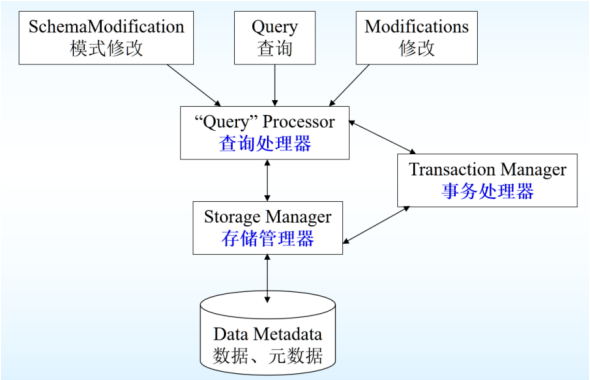

图 2: 数据库系统的结构

3. 数据库系统（DataBase System 简称 DBS）引入数据库后的计算机系统

1.2 数据模型

数据模型是现实世界数据特征的抽象，通俗地讲，数据模型就是现实世界的模拟

1. 计算机只能处理数字化的数据，需要使用数据模型来抽象、表示和处理现实世界中的具体事物。

2. 现有的数据库系统均是基于某种数据模型的。数据模型是数据库系统的核心和基础。

3. 数据模型应满足三个要求：能比较真实地模拟现实世界；容易为人所理解；便于在计算机上实现。

数据模型分为两类：

1. 概念模型（信息模型）按用户的观点对数据库进行设计，主要用于数据库设计，**与 DBMS 无关**

2. DBMS 支持的数据模型

1）逻辑模型，按计算机系统的观点对数据建模，用于 DBMS 实现。包括层次模型、网状模型、关系模型、面向对象模型、对象关系模型等

2）物理模型，是对数据最底层的抽象，描述数据在系统内部的表示方式和存取方法，在磁盘或磁带上的存储结构和存取方法。它的具体实现是 DBMS 的任务。

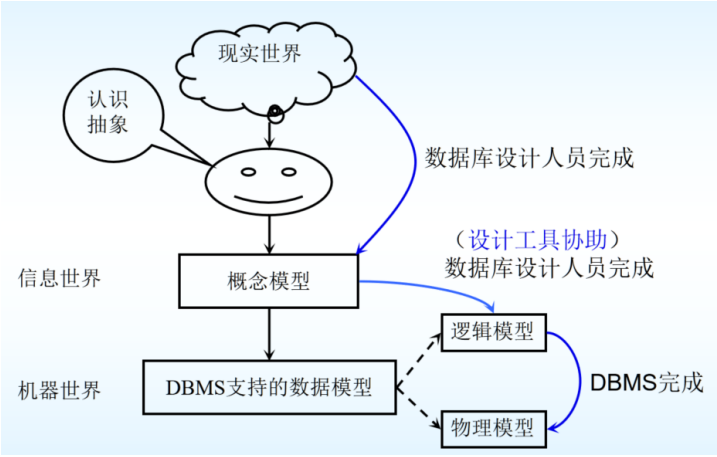

图 3: 现实世界中客观对象的抽象过程

其中最难的部分是从现实世界抽象到概念模型的过程
数据模型精确描述了系统的静态特性、动态特性和完整性约素条件。
数据模型三要素：

1. 数据结构（静态）

   数据库组成对象以及对象之间的关系

2. 数据操纵（动态）

   数据库中各种对象的实例允许执行操作的集合，包括操作及有关的操作规则

   主要操作：查询、更新（插入、删除、修改）——增删查改

3. 数据完整约束

   数据及其联系所具有的制约和依存的一组规则，也称为完整性规则

   限定数据库状态以及状态的变化，以保证数据的正确、有效、相容

1.2.1 概念模型

概念模型是信息建模，是对现实世界的事物符号化的描述，为计算机处理做准备。常用的概念模型是**实体联系图**）（Entity-Relationship Diagram, E-R 图）

信息世界的主要概念：

1. 实体（Entity）客观存在的各类事物
2. 属性（Attribute）实体所具有的特性
3. 码（Key）能唯一标识实体的属性集
4. 域（Domain）属性的取值范围
5. 实体型（Entity Type）对具有相同属性特征实体的描述
6. 实体集（Entity Set）同型实体的集合
7. 联系（Relationship）不同实体集中实体之间的联系，也可以是同一实体集内实体的联系

联系的种类：

1. 一对一联系 (1:1)
2. 一对多联系 (1:N)
3. 多对多联系 (M:N)

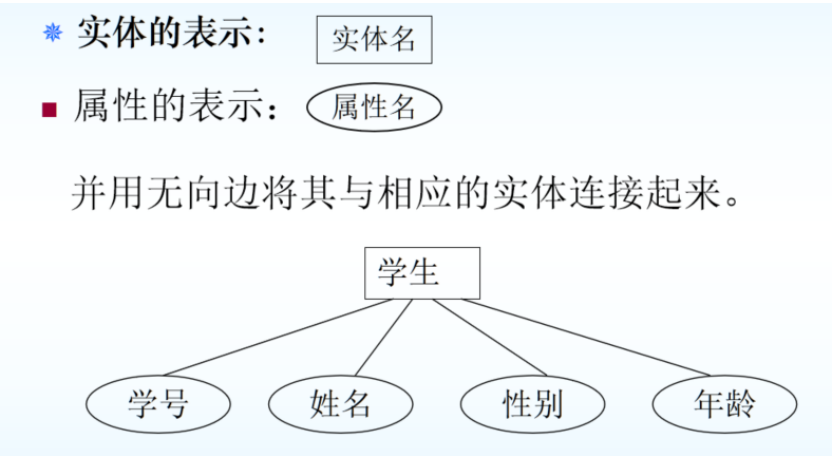

图 4: 用 E-R 图建立概念模型

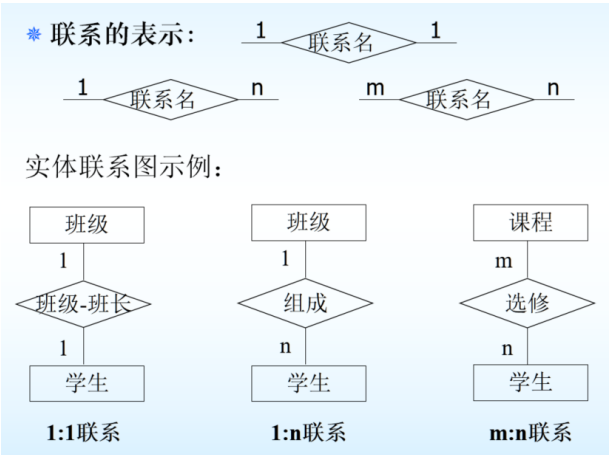

图 5: 联系的表示

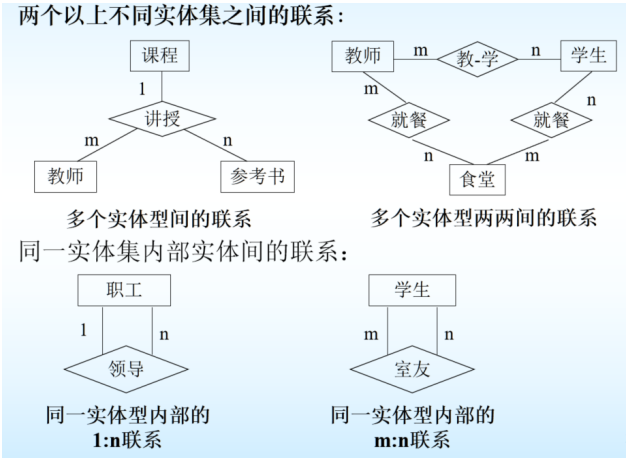

图 6: 实体集的联系

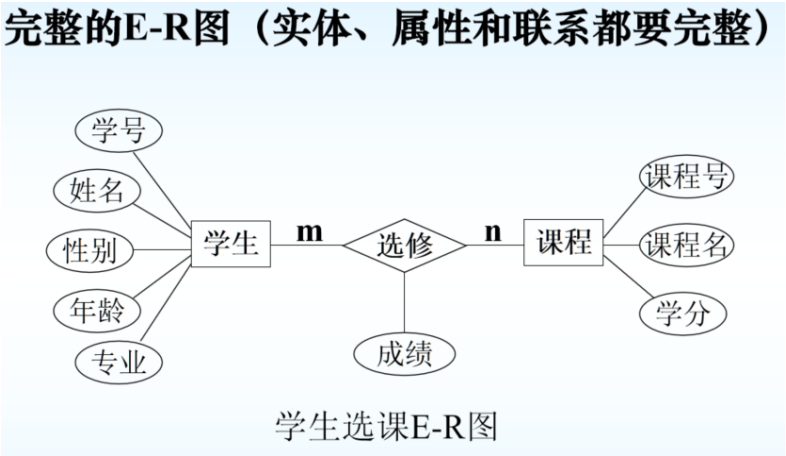

图 7: 完整 E-R 图

超类与子类：

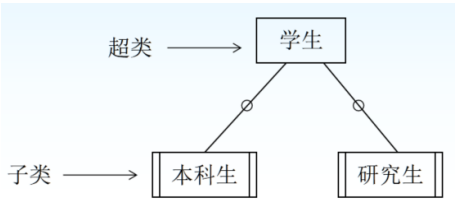

图 8: 超类与子类

特殊化（specialization）从实体集中找出与其他实体具有不同属性集的子集构成新实体的过程

一般化（generalization）从实体集中提取公共属性构成新实体的过程

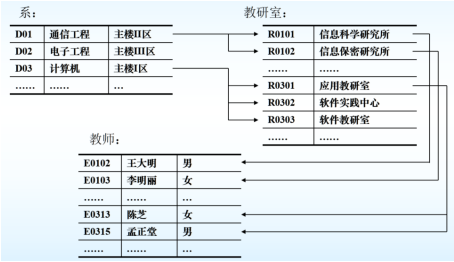

图 10: 层次模型实现示意图

1.2.2 逻辑模型

1. 层次模型（Hierarchical Model）

   数据结构：树型结构，每个结点表示一个记录类型（包含若干属性），每条边表示一个记录类型之间的一对多关系。

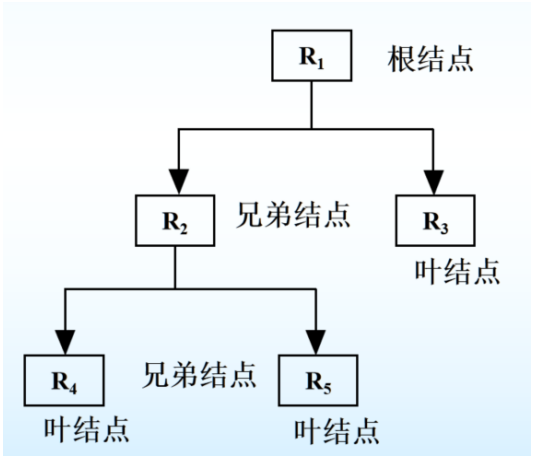

图 9: 层次模型

满足如下两个条件：

(a) 有且只有一个结点没有父结点，称为根节点

(b) 根节点以外的其他结点有且只有一个父结点

特点：

(a) 结点的父结点是唯一的

(b) 只能一对多或一对一

(c) 每个记录类型可以定义一个排序字段，也称码字段

(d) 任何记录值只有按其路径查看，才能显出全部意义

(e) 没有一个子结点记录值能脱离父结点记录值而独立存在

多对多在层次模型中的表示——分解成一对多的联系

分解方法：

(a) 冗余节点法

冗余结点法可以随意改变结点的存储位置，但增加了额外存储空间，容易造成数据的不一致性。

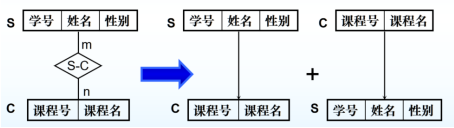

图 11: 冗余节点法

(b) 虚拟节点法

虚拟结点法改变结点存储位置后必须修改虚拟结点的指针，但不会产生数据的不一致性。

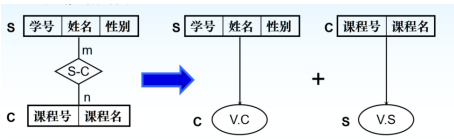

图 12: 虚拟节点法

数据操纵与约束：

(a) 操纵有增删查改

(b) 查询要从根结点出发做树的遍历

(c) 插入时、若无父结点则无法插入子结点值

(d) 删除时必须删除其子树

(e) 修改必须保证数据一致性

存储结构：

(a) 邻接表法

(b) 链接法

优点：

(a) 模型简单，对具有一对多的层次关系的部门描述自然、客观，容易理解

(b) 性能优于关系模型，不低于网状模型

(c) 层次数据模型提供了良好的完整性支持

缺点：

(a) 多对多表达不自然

(b) 对插入和删除限制多

(c) 查询结点必须通过父结点

(d) 层次命令趋于程序化

2. 网状模型

记录类型之间的联系用连线表达，数据结构是网，联系必须标注名称

数据操纵和完整性约束

(a) 操纵同层次模型

(b) 支持码（对应实体的标识符）

(c) 对于每条边，父结点和子结点有一对多的联系

(d) 支持父结点与子结点间的某些约束

存储结构使用链式存储

优点:

(a) 更直观的描述现实世界

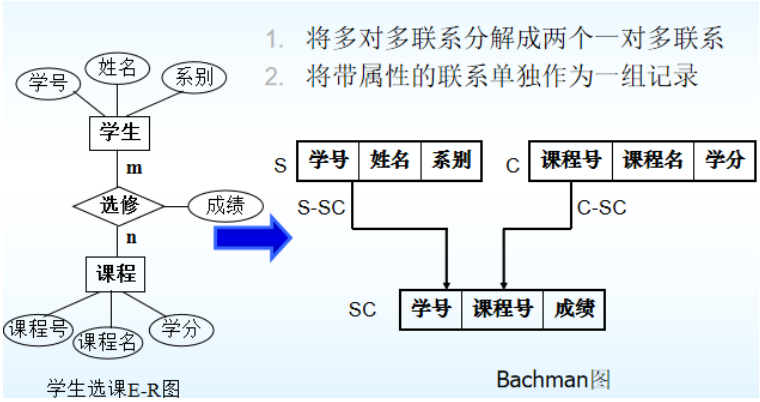

图 13: 网状模型的多对多联系

(b) 具有良好的性能，存取效率较高

缺点:

(a) 结构比较复杂，随着应用环境扩大，数据库的结构解越来越复杂

(b) 网状模型的 DDL、DML 语言复杂，用户不容易使用

3. 关系模型

数据操纵与完整性约束

(a) 查询、插入、删除、修改。数据操作是集合操作，操作对象和操作结果都是关系，即若干元组的集合

(b) 实体完整性、参照完整性、用户定义完整性

存储结构：实体和联系都是用关系来表示，存储在数据文件中

(a) 实体型直接用关系表示

(b) 一对一关系：可隐含在实体对应的关系中

(c) 一对多关系：可隐含在实体对应的关系中

(d) 多对多关系：直接用关系表示

优点

(a) 建立在严格的数据概念基础上

(b) 概念单一，数据结构简单、清晰，用户易懂易用（实体和各类联系都用关系表示，对数据的检索也是关系）

(c) 关系的存取路径对用户透明（具有更高的数据独立性，更好的安全保密性。简化了程序员的工作）

缺点

(a) 存取路径对用户透明导致查询效率不如其他逻辑模型。提高性能必须对用户的查询请求进行优化，增加了开放数据库管理系统的难度。

4. 面向对象模型

5. 对象关系模型

1.3 数据库系统结构

划分方法：从用户的角度看

1. 集中式结构

2. 分布式结构

3. 客户/服务器结构

4. 浏览器/应用服务器/数据库服务器多层结构

从数据库管理系统角度看：**三级模式结构**
数据库系统模式

1. 型（Type）：对某一类数据的结构和属性说明

2. 值（Value）：型的某一个具体值

模式：（Schema）

1. 数据库逻辑结构和特征的描述

2. 型的描述

3. 反映数据的结构及其联系

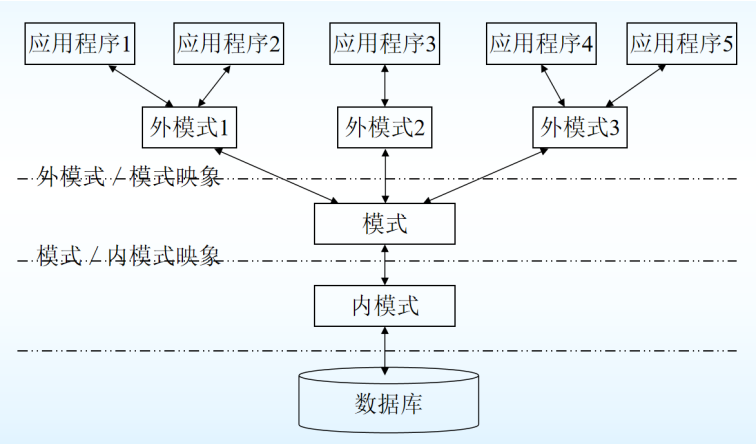

图 14: 数据库的三级模式结构

4. 模式时相对稳定的

模式的一个实例（Instance）

1. 模式的一个具体值
2. 反映数据库某一时刻的状态
3. 同一个模式有很多实例
4. 实例随数据库中的数据的更新而变动

模式：（Schema 也称逻辑模式）

1. 数据库中全体数据的逻辑结构和特征描述
2. 所有用户的公共数据视图，综合所有用户的需求

一个数据库只有一个模式
模式的定位：是数据库模式结构的中间层

1. 与数据库的物理存储细节和硬件关系无关
2. 与具体的应用程序、开放工具及高级程序设计语言无关

模式的定义

1. 数据的逻辑结构（数据项的名字、类型、取指范围等）

2. 数据之间的联系

3. 数据有关的安全性、完整性要求

外模式（External Schema，也称子模式或用户模式）

1. 数据库用户使用局部数据的逻辑结构和特征描述

2. 数据库用户的数据视图，是与某一应用有关的数据逻辑表示

外模式的地位：介于模式与应用之间
模式与外模式的关系：一对多

1. 外模式通常是模式的子集

2. 一个数据库可以有多个外模式。反映了不同用户的应用需求、看待数据的方式。对外数据保密的要求

3. 对模式中同一数据，在外模式中的结构、类型、长度、保密等级都可以不同

4. 外模式中可能有更多的信息

外模式与应用的关系：一对多

1. 同一外模式可以为某一用户的多个应用系统使用

2. 一个应用程序只能使用一个外模式

外模式的用途：保证数据库安全的一个有力措施：每个用户只能看见和访问所对应的外模式中的数据

内模式（Internal Schema 也称存储模式）

1. 数据物理结构和存储方式的描述

2. 是数据在数据库内部的表示方式，包括

   (a) 记录的存储方式（顺序存储、按照 B 树结构存储、按 Hash 方法存储）

   (b) 索引的组织方式

   (c) 数据是否压缩

   (d) 数据是否加密

   (e) 关于数据存储记录结构的规定（定长/变长结构等）

一个数据只有一个内模式