数 据 库 原 理

Theory of Database

李静

信息科学与技术学院

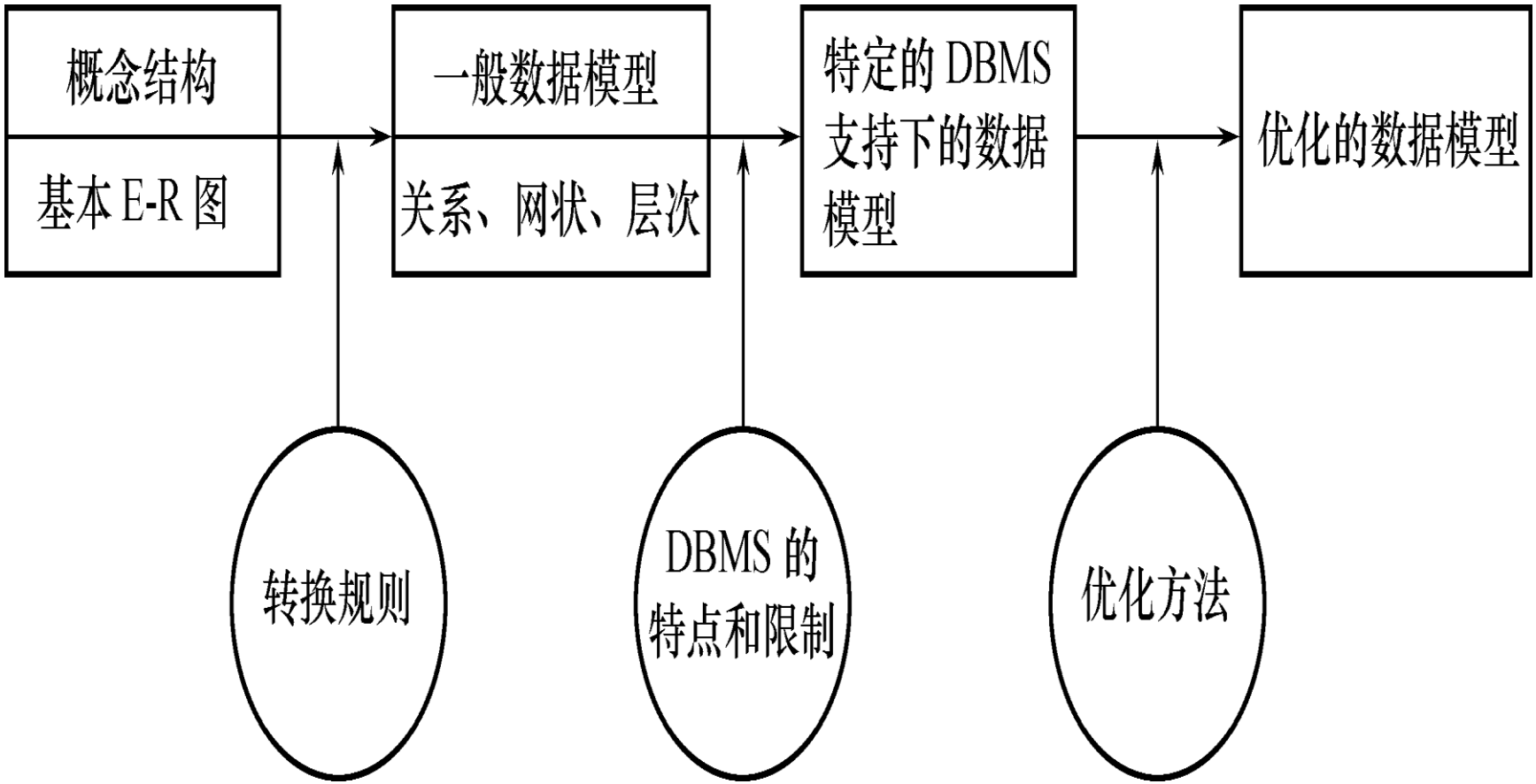
第8章 数据库设计

- 8.1 数据库设计概述
- 8.2 数据库需求分析
- 8.3 数据库结构设计
- 8.4 数据库行为设计
- 8.5 数据库实施

8.3.2 逻辑结构设计

- 把概念结构设计阶段设计好的基本E-R模型转换为具体的数据库管理系统支持的数据模型。
- 步骤：
  - 将概念模型转换为某种组织层数据模型；
  - 对数据模型进行优化。

逻辑结构设计的步骤


概念结构
基本 E-R 图
一般数据模型
关系、网状、层次
特定的 DBMS 支持下的数据模型
优化的数据模型
转换规则
DBMS 的特点和限制
优化方法

E-R图向关系模型的转换

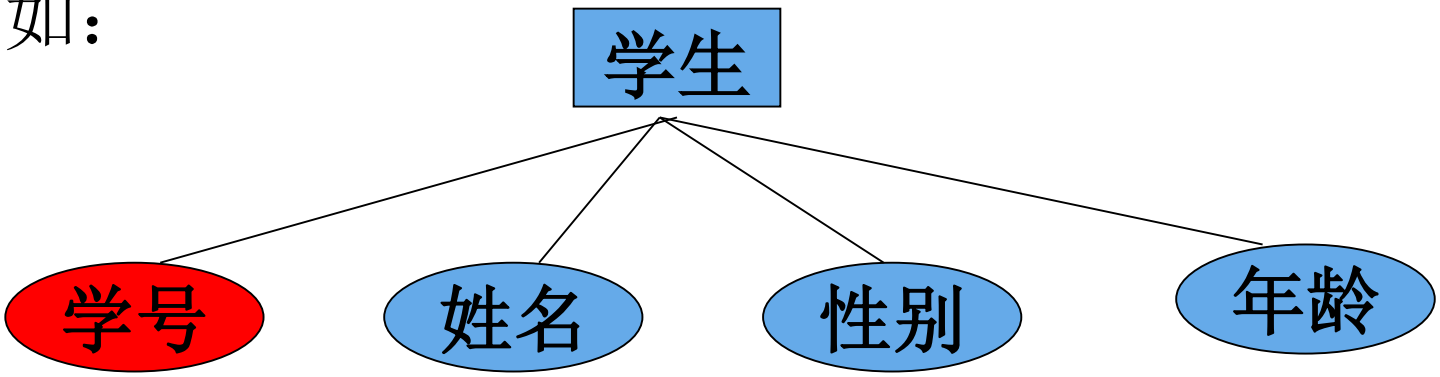
- 转换内容:
  - 将实体型和实体间的联系转换为关系模式
  - 确定这些关系模式的属性和码

实体的转换原则

- 实体的转换原则
  - 一个实体转换为一个关系模式；
  - 实体的名称对应关系模式的名称；
  - 实体的属性对应关系模式的属性；
  - 实体的候选码对应关系模式的候选码。

实体的转换原则

- 例如：



- 转换为关系模式：

学生（学号，姓名，性别，年龄）

联系的转换原则

(1)一个1:1联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。

- 转换为一个独立的关系模式：

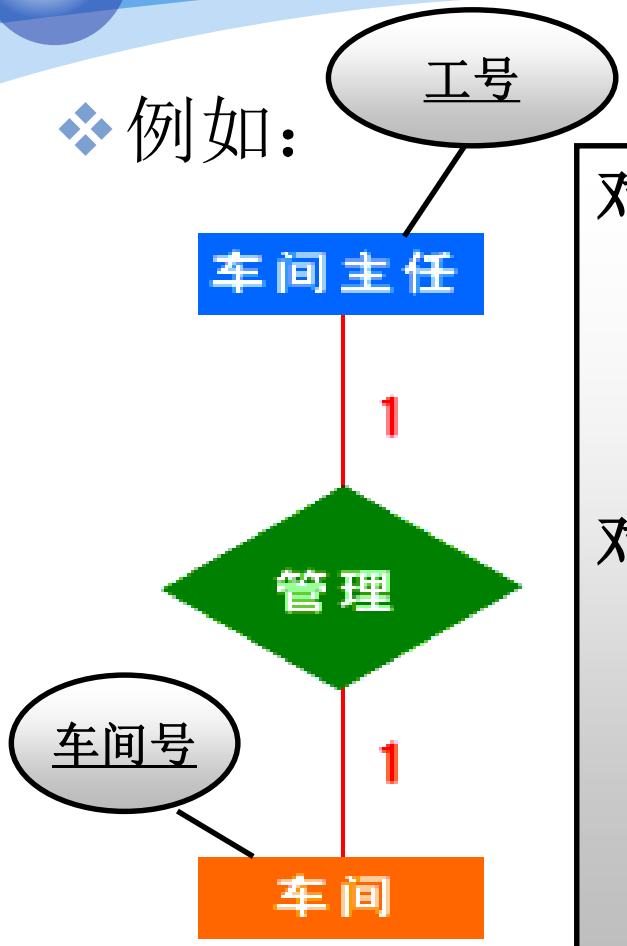
  相连实体的码、联系的属性——〉新关系的属性， 每个实体的码均是该关系的候选码。

- 与某一端实体对应的关系模式合并：

  需在该关系模式的属性中加入另一实体的码和联系的属性。

联系的转换原则

❖ 例如：



对实体的转换：

车间主任（工号，.....）

车间（车间号，.....）

对联系的转换：

合并到其中的一个实体关系中。

车间主任（工号，车间号.....）

新关系：管理（工号，车间号，.....）

联系的转换原则

(2)一个1:n联系可以转换为一个独立的关系模式，也可以与n端对应的关系模式合并。

- 转换为一个独立的关系模式：

  相连实体的码、联系的属性——〉新关系的属性，n端实体的码——〉新关系的码。

- 与n端对应的关系模式合并

  在n端的实体表中增加1端实体的候选码。

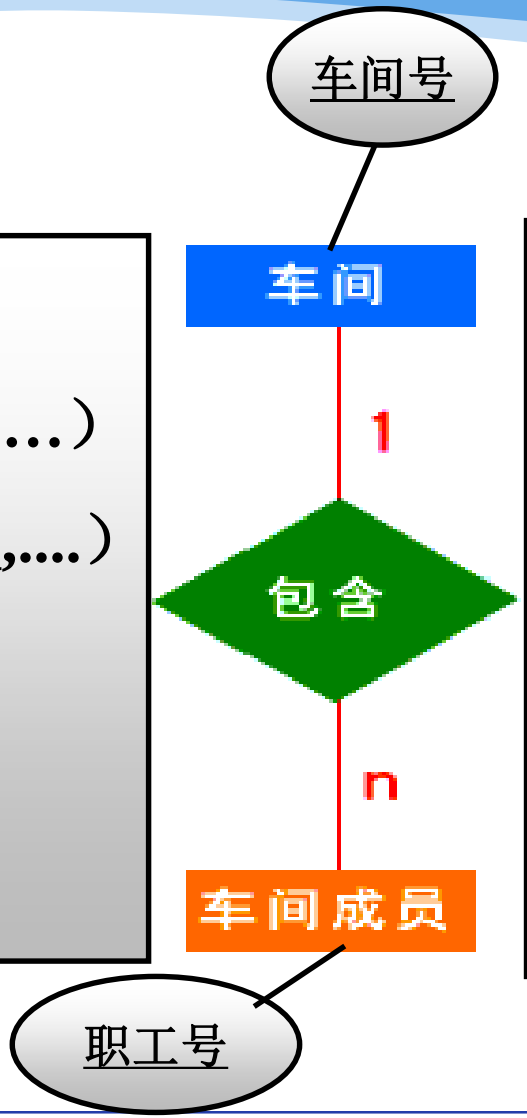
联系的转换原则

- 例如：

对实体的转换：

车间（车间号，……）

车间成员（职工号,....）



对联系的转换：

新关系模式：

车间组成（职工号，车间号…）

合并到n端实体关系中：

车间成员（职工号，车间号……）

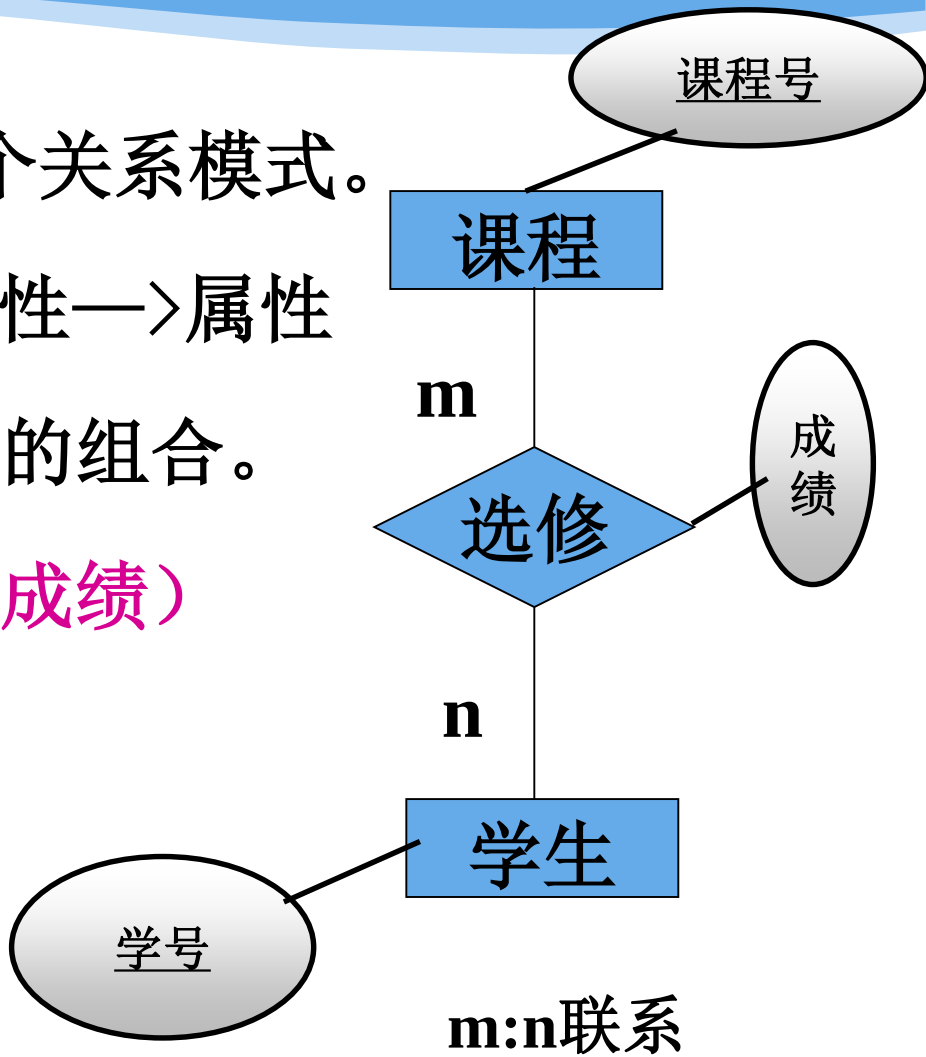
联系的转换原则

(3) 一个m:n联系转换为一个关系模式。

相连实体的码、联系的属性—>属性

新关系的码是各实体码的组合。

选修（学号，课程号，成绩）

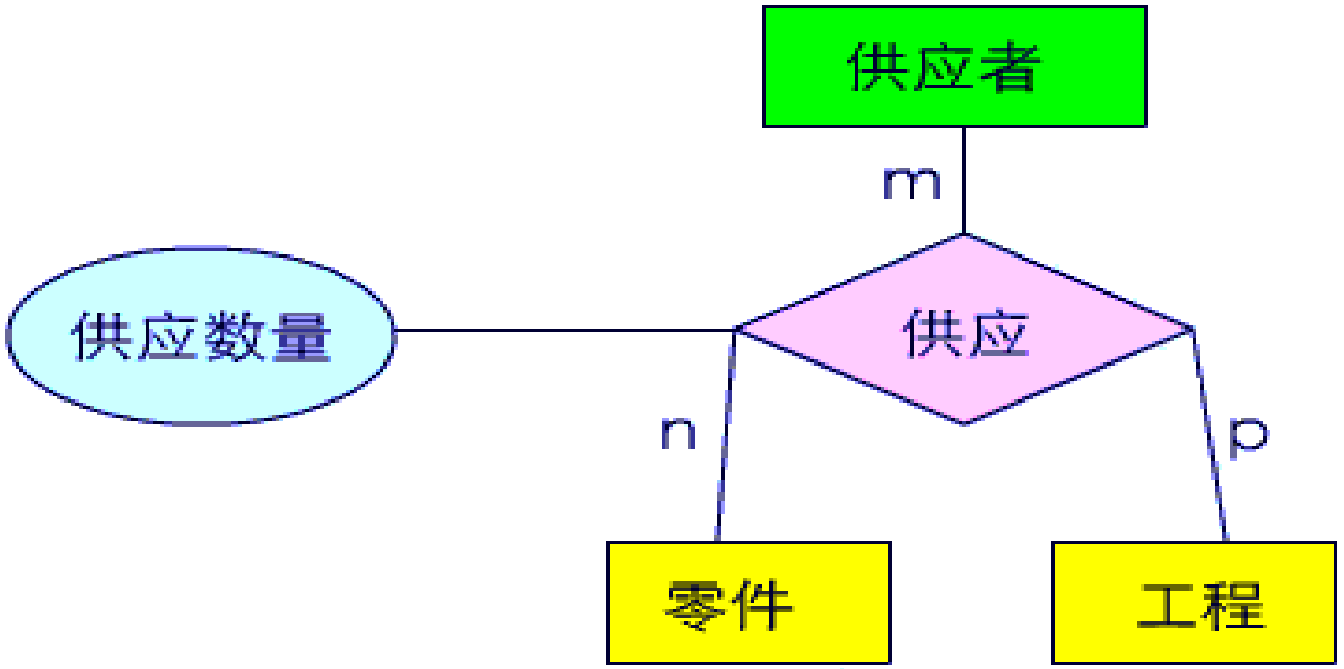


联系的转换原则

(4) 三个或以上实体间的多元联系转换新关系模式。

例如：

工程供应零件表（供应者号，零件号，工程号，供应数量）



转换原则

(5)具有相同码的关系模式可合并

目的：减少系统中的关系个数

合并方法：

将其中一个关系模式的全部属性加入到另一个关系模式中，然后去掉其中的同义属性（可能同名也可能不同名），并适当调整属性的次序

转换原则

注意：

- 从理论上讲，1:1联系可以与任意一端对应的关系模式合并
- 但在一些情况下，与不同的关系模式合并效率会大不一样。因此究竟应该与哪端的关系模式合并需要依应用的具体情况而定。
- 由于连接操作是最费时的操作，所以一般应以尽量减少连接操作为目标。

  例如，如果经常要查询某个班级的班主任姓名，则将管理联系与教师关系合并更好些。

数据模型的优化

- 得到初步数据模型后，还应该适当地修改、调整数据模型的结构，以进一步提高数据库应用系统的性能，这就是数据模型的优化。

- 关系数据模型的优化通常以规范化理论为指导。

数据模型的优化的方法

1、确定数据依赖

关系模式内部各属性之间的数据依赖

不同关系模式属性之间数据依赖

2、消除冗余的联系

3、确定所属范式

4、按照数据处理的要求，确定是否合并或分解。

注意：并不是规范化程度越高的关系就越优，一般说来，第三范式就足够了

数据模型的优化

例：关系模式

学生成绩单(学号,英语,数学,语文,平均成绩)

存在下列函数依赖：

学号→英语　学号→数学　学号→语文

学号→平均成绩　(英语, 数学, 语文)→平均成绩

显然有：

学号→(英语,数学,语文)　R∈2NF

平均成绩要不要保留？

常用分解方法

通过对关系模式进行必要的分解，提高数据操作的效率和存储空间的利用率。

- 常用分解方法
  - 水平分解
  - 垂直分解

常用分解方法

- 水平分解
  - ➢什么是水平分解
    - 把(基本)关系的元组分为若干子集合，定义每个子集合为一个子关系，以提高系统的效率
  - ➢水平分解的适用范围
    - 满足“80/20原则”的应用
    - 并发事务经常存取不相交的数据

常用分解方法

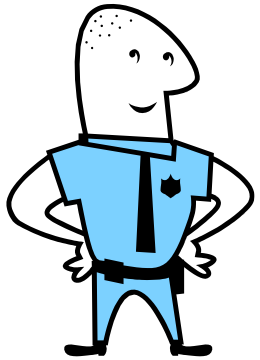
- 垂直分解
  - ➢什么是垂直分解
    - –把关系模式*R*的属性分解为若干子集合，形成若干子关系模式
  - ➢垂直分解的适用范围
    - –取决于分解后*R*上的所有事务的总效率是否得到了提高

设计外模式

- 将概念模型转换为逻辑数据模型之后，还应该根据局部应用需求，并结合具体的数据库管理系统的特点，设计用户的外模式。
- 外模式概念对应关系数据库的视图概念，设计外模式是为了更好地满足局部用户需求。
- 定义数据库的模式主要是从系统的时间效率、空间效率、易维护等角度出发。

定义外模式考虑事项

- 使用更符合用户习惯的别名。
- 对不同级别的用户定义不同的视图，以保证数据的安全。
- 简化用户对系统的使用。



定义用户外模式——例子

关系模式产品（产品号，产品名，规格，单价，生产车间，生产负责人，产品成本，产品合格率，质量等级）

- 只允许顾客查询的属性
- 只允许销售部门查询的属性
- 生产领导部门则可以查询全部产品数据
- 防止用户非法访问数据，保证系统的安全性

可以在产品关系上建立两个视图：

为一般顾客建立视图：

产品1（产品号，产品名，规格，单价）

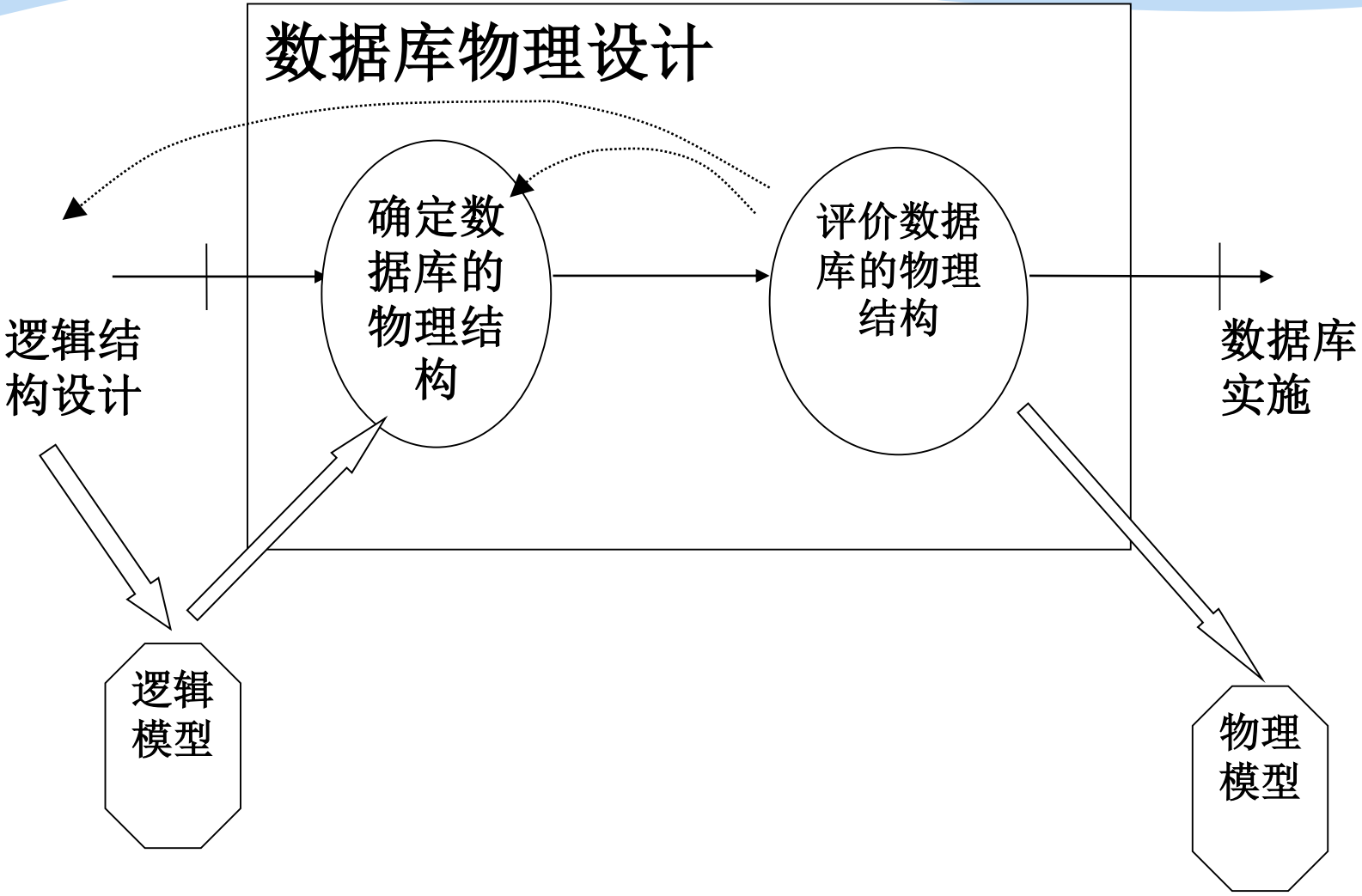
为产品销售部门建立视图：

产品2（产品号，产品名，规格，单价，车间，生产负责人）

8.3.3 物理结构设计

- 数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，依赖于选定的数据库管理系统.
- 为一个给定的逻辑数据模型选取一个最适合应用环境的物理结构的过程，就是数据库的物理设计.

数据库的物理设计(续)



物理结构设计的内容和方法

- 设计物理数据库结构的准备工作
  - 对要运行的事务进行详细分析，获得选择物理数据库设计所需参数
  - 充分了解所用RDBMS的内部特征，特别是系统提供的存取方法和存储结构

选择物理数据库设计所需参数

- 数据库查询事务
  - 查询的关系
  - 查询条件所涉及的属性
  - 连接条件所涉及的属性
  - 查询的投影属性
- 数据更新事务
  - 被更新的关系
  - 每个关系上的更新操作条件所涉及的属性
  - 修改操作要改变的属性值
- 每个事务在各关系上运行的频率和性能要求

关系数据库物理设计的内容

- 为关系模式选择存取方法(建立存取路径)
- 设计关系、索引等数据库文件的物理存储结构

关系模式存取方法选择

- 数据库系统是多用户共享的系统，对同一个关系要建立多条存取路径才能满足多用户的多种应用要求
- 物理设计的任务之一就是要确定选择哪些存取方法，即建立哪些存取路径

关系模式存取方法选择（续）

- **DBMS常用存取方法**
  - 索引方法
    - 目前主要是B+树索引方法
    - 经典存取方法，使用最普遍
  - 聚簇（Cluster）方法
  - HASH方法

一、索引存取方法的选择

- 根据应用要求确定
  - 对哪些属性列建立索引
  - 对哪些属性列建立组合索引
  - 对哪些索引要设计为唯一索引

索引存取方法的选择（续）

- 选择索引存取方法的一般规则
  - 属性经常在查询条件中出现
  - 属性经常作为最大值和最小值等聚集函数的参数
  - 属性经常在连接操作的连接条件中出现
- 关系上定义的索引过多会带来较多的额外开销
  - 维护索引的开销
  - 查找索引的开销

二、聚簇存取方法的选择

❖聚簇

- 为了提高某个属性的查询速度，把这个属性（称为聚簇码）上具有相同值的元组集中存放在连续的物理块称为聚簇

聚簇存取方法的选择（续）

❖聚簇的用途

- **1. 大大提高按聚簇码进行查询的效率**

例：假设学生关系按所在系建有索引，现在要查询信息系的所有学生名单。

➢信息系的500名学生分布在500个不同的物理块上时，至少要执行500次I/O操作

➢如果将同一系的学生元组集中存放，则每读一个物理块可得到多个满足查询条件的元组，从而显著地减少了访问磁盘的次数

聚簇存取方法的选择（续）

- **2.** 节省存储空间
  - 聚簇以后，聚簇码相同的元组集中在一起了，因而聚簇码值不必在每个元组中重复存储，只要在一组中存一次就行了

聚簇存取方法的选择（续）

- 聚簇的局限性
  - 1. 聚簇只能提高某些特定应用的性能
  - 2. 建立与维护聚簇的开销相当大
    - 对已有关系建立聚簇，将导致关系中元组移动其物理存储位置，并使此关系上原有的索引无效，必须重建
    - 当一个元组的聚簇码改变时，该元组的存储位置也要做相应移动

聚簇存取方法的选择（续）

- 聚簇的适用范围

1. 既适用于单个关系独立聚簇，也适用于多个关系组合聚簇

例：假设用户经常要按系别查询学生成绩单，这一查询涉及学生关系和选修关系的连接操作，即需要按学号连接这两个关系，为提高连接操作的效率，可以把具有相同学号值的学生元组和选修元组在物理上聚簇在一起。这就相当于把多个关系按“预连接”的形式存放，从而大大提高连接操作的效率。

聚簇存取方法的选择（续）

2. 当通过聚簇码进行访问或连接是该关系的主要应用，与聚簇码无关的其他访问很少或者是次要的时，可以使用聚簇。

- 尤其当SQL语句中包含有与聚簇码有关的ORDER BY，GROUP BY，UNION，DISTINCT等子句或短语时，使用聚簇特别有利，可省去对结果集的排序操作

聚簇存取方法的选择（续）

- 设计候选聚簇
  - 对经常在一起进行连接操作的关系可以建立聚簇
  - 如果一个关系的一组属性经常出现在相等比较条件中，则该单个关系可建立聚簇
  - 如果一个关系的一个(或一组)属性上的值重复率很高，则此单个关系可建立聚簇。即对应每个聚簇码值的平均元组数不太少。太少了，聚簇的效果不明显

聚簇存取方法的选择（续）

- 优化聚簇设计
  - 经常进行全表扫描的关系不建立聚簇；
  - 更新操作远多于连接操作的关系不建立聚簇；
  - 不同的聚簇中可能包含相同的关系，一个关系可以在某一个聚簇中，但不能同时加入多个聚簇

  从这多个聚簇方案(包括不建立聚簇)中选择一个较优的，即在这个聚簇上运行各种事务的总代价最小

三、HASH存取方法的选择

❖选择HASH存取方法的规则

- 当一个关系满足下列两个条件时，可选择HASH方法
  - ➢该关系的属性主要出现在等值连接条件中或主要出现在相等比较选择条件中
  - ➢该关系的大小可预知，而且不变；或该关系的大小动态改变，但所选用的DBMS提供了动态HASH存取方法

确定数据库的存储结构

- 确定数据库物理结构的内容
  - 1. 确定数据的存放位置和存储结构
    - 关系
    - 索引
    - 聚簇
    - 日志
    - 备份
  - 2. 确定系统配置

1. 确定数据的存放位置

- 确定数据存放位置和存储结构的因素
  - 存取时间
  - 存储空间利用率
  - 维护代价

这三个方面常常是相互矛盾的

例：消除一切冗余数据虽能够节约存储空间和减少维护代价，但往往会导致检索代价的增加

必须进行权衡，选择一个折中方案

确定数据的存放位置（续）

- 基本原则
  - 根据应用情况将
    - 易变部分与稳定部分分开存放
    - 存取频率较高部分与存取频率较低部分，分开存放

确定数据的存放位置（续）

例：

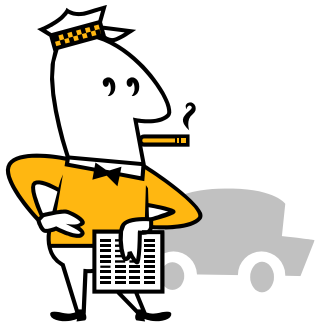
- **数据备份、日志文件备份等由于只**在故障恢复时才使用，而且数据量很大，可以考虑单独存放
- 如果计算机有多个磁盘，可以考虑将表和索引分别放在不同的磁盘上，在查询时，由于磁盘驱动器并行工作，可以提高物理I/O读写的效率
- 可以将比较大的表分别放在两个磁盘上，以加快存取速度，这在多用户环境下特别有效
- 可以将日志文件与数据库对象（表、索引等）放在不同的磁盘以改进系统的性能

2. 确定系统配置

- DBMS产品一般都提供了一些存储分配参数
  - 同时使用数据库的用户数
  - 同时打开的数据库对象数
  - 内存分配参数
  - 使用的缓冲区长度、个数
  - 存储分配参数
  - ……

物理结构设计的评价

- 评价物理结构设计的方法完全依赖于具体的DBMS，主要考虑的是操作开销，即为使用户获得及时、准确的数据所需的开销和计算机的资源的开销。具体可分为如下几类：
  - 查询和响应时间
  - 更新事务的开销
  - 生成报告的开销
  - 主存储空间的开销
  - 辅助存储空间的开销



小结

- 需求分析
- 结构设计

概念结构设计
逻辑结构设计
物理结构设计

8.4 数据库行为设计

- 8.4.1 功能需求分析
- 8.4.2 功能设计
- 8.4.3 事务设计

8.4.1 功能需求分析

- 在进行需求分析时，实际上进行了两项工作:
  - “数据流”的调查分析，
  - “事务处理”过程的调查分析。
- 数据流的调查分析为数据库的信息结构提供了最原始的依据，
- 事务处理的调查分析是行为设计的基础。

对行为特性要进行的分析

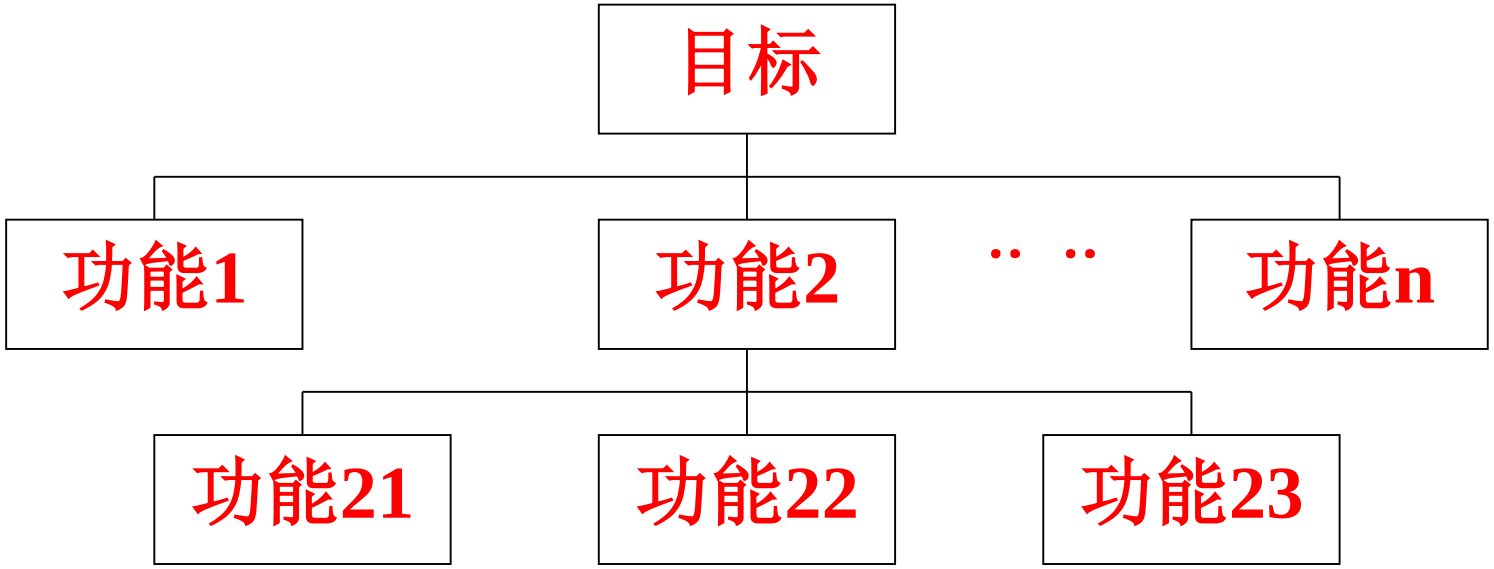
- 标识所有的查询、报表、事务及动态特性，指出对数据库所要进行的各种处理；
- 对每个实体所进行的操作（增、删、改、查）；
- 给出每个操作的语义，包括结构约束和操作约束；
- 给出每个操作（针对某一对象）的频率；
- 给出每个操作（针对某一应用）的响应时间；
- 给出该系统总的目标。

示例

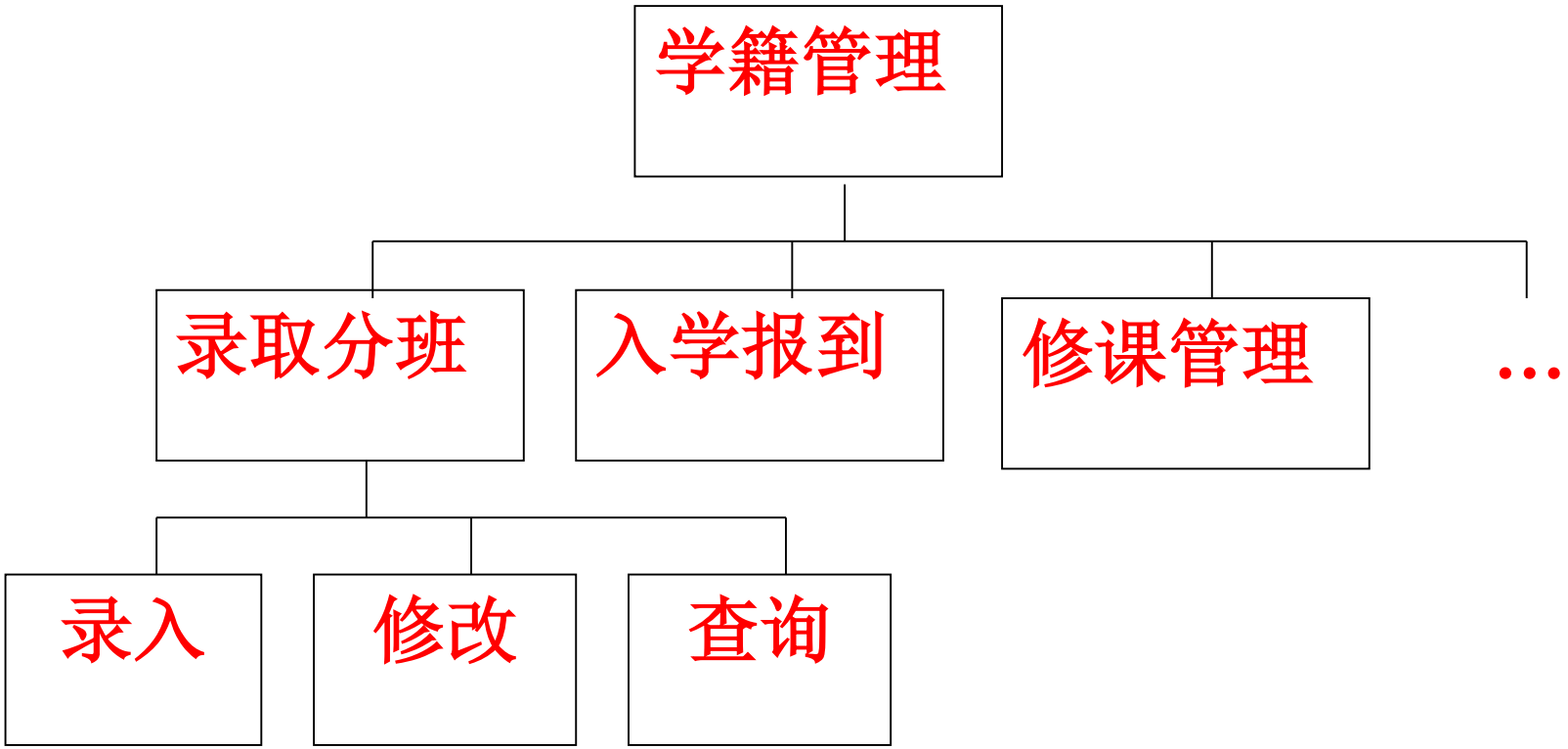
- 教师退休行为的操作特征为：
  - 该教师没有未教授完的课程。
  - 删除此教师记录。
  - 此教师记录不再在当前教师表中。

8.4.2 功能设计

系统目标的实现是通过系统的各功能模块来达到的。由于每个系统功能又可以划分为若干个更具体的功能模块，因此，可以从目标开始，一层一层分解下去，直到每个子功能模块只执行一个具体的任务。



例：“学籍管理”的功能结构图

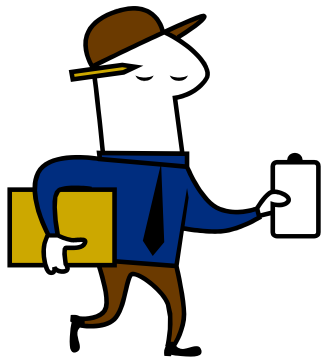


8.4.3事务设计

- 事务处理是计算机模拟人处理事务的过程，包括：
  - 输入设计
  - 输出设计
  - 功能设计
  - 等等

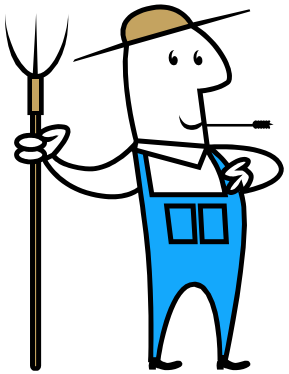
输入设计

- 原始单据的设计格式
- 制成输入一览表
- 制作输入数据描述文档



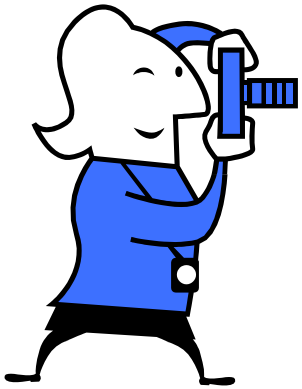
输出设计

- 用途。区分输出结果是给客户的还是用于内部或报送上级领导的。
- 输出设备的选择。是仅仅显示出来，还是要打印出来或需要永久保存。
- 输出量。
- 输出格式。



8.5 数据库实施

- 加载数据
- 调试和运行应用程序



加载数据

- 在数据库系统中，一般数据量都很大，各应用环境差异也很大 。
- 为了保证数据库中的数据正确、无误，必须十分重视数据的校验工作。
- 在将数据输入系统进行数据转换过程中，应该进行多次的校验。
- 对于重要的数据的校验更应该反复多次，确认无误后再进入到数据库中。

调试和运行应用程序

- 在有一部分数据加载到数据库之后，就可以开始对数据库系统进行联合调试了，这个过程又称为数据库试运行。
- 这一阶段要实际运行数据库应用程序，执行对数据库的各种操作，测试应用程序的功能是否满足设计要求。如果不满足，则要对应用程序进行修改、调整，直到达到设计要求为止。
- 在数据库试运行阶段，还要对系统的性能指标进行测试，分析其是否达到设计目标。

数据库的试运行

- 在原有系统的数据有一小部分已输入数据库后，就可以开始对数据库系统进行联合调试，称为数据库的试运行
- 数据库试运行主要工作包括：

1）功能测试

  - 实际运行数据库应用程序，执行对数据库的各种操作，测试应用程序的功能是否满足设计要求

2）性能测试

  - 测量系统的性能指标，分析是否达到设计目标

数据库的试运行（续）

强调两点：

- 分期分批组织数据入库
  - 重新设计物理结构/逻辑结构，需数据重新入库
  - 由于数据入库工作量实在太大，费时、费力，所以应分期分批地组织数据入库
  - 先输入小批量数据供调试用
    - 待试运行基本合格后再大批量输入数据
    - 逐步增加数据量，逐步完成运行评价

数据库的试运行（续）

- 数据库的转储和恢复
  - 在数据库试运行阶段，系统还不稳定，硬/软件故障随时都可能发生
  - 系统的操作人员对新系统还不熟悉，误操作也不可避免
  - 因此必须做好数据库的转储和恢复工作，尽量减少对数据库的破坏。

8.6 数据库的运行和维护

- 数据库投入运行标志着开发工作的基本完成和维护工作的开始，数据库只要存在一天，就需要不断地对它进行评价、调整和维护。
- 在数据库运行阶段，对数据库的经常性的维护工作主要由数据库系统管理员完成，其主要工作包括：
  - 数据库的备份和恢复
  - 数据库的安全性和完整性控制
  - 监视、分析、调整数据库性能
  - 数据库的重组

数据库的备份和恢复

- 要对数据库进行定期的备份，一旦出现故障，要能及时地将数据库恢复到尽可能的正确状态，以减少数据库损失。

数据库的安全性和完整性控制

- 随着数据库应用环境的变化，对数据库的安全性和完整性要求也会发生变化。如：
  - 收回某些用户的权限，
  - 增加、修改某些用户的权限，
  - 增加、删除用户，
  - 数据的取值范围发生变化等。
- 这都需要系统管理员对数据库进行适当的调整，以反映这些新的变化。

监视、分析、调整数据库性能

- 监视数据库的运行情况，并对检测数据进行分析，找出能够提高性能的可行性，并适当地对数据库进行调整。
- 目前有些DBMS产品提供了性能检测工具，数据库系统管理员可以利用这些工具很方便地监视数据库。

数据库的重组

- 数据库经过一段时间的运行后，随着数据的不断添加、删除和修改，会使数据库的存取效率降低，数据库管理员可以改变数据库数据的组织方式，
- 通过增加、删除或调整部分索引等方法，改善系统的性能。
- 数据库的重组并不改变数据库的逻辑结构。

总　结

- 数据库设计概述
- 数据库需求分析
- 数据库结构设计
- 数据库行为设计
- 数据库实施

作业

📖 P141 8、9、10