数据库系统概论

An Introduction to Database System

第七章 数据库设计(续2)

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

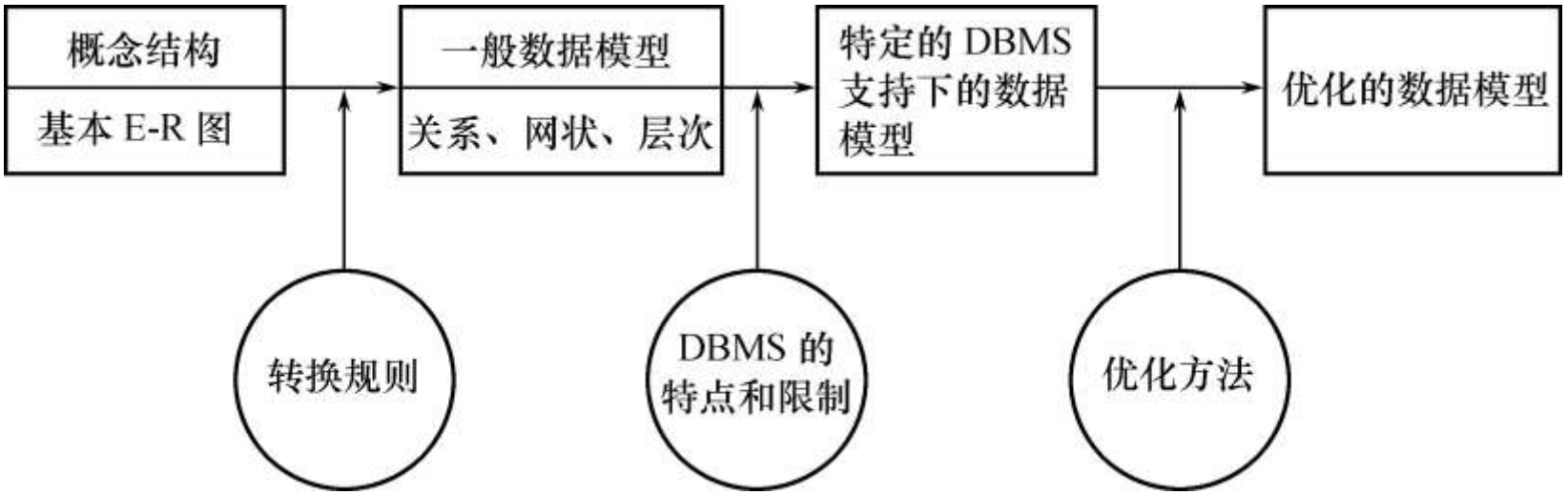
7.6 数据库的实施和维护

7.7 小结

7.4 逻辑结构设计

- 逻辑结构设计的任务
  - 把概念结构设计阶段设计好的基本E-R图转换为与选用DBMS产品所支持的数据模型相符合的逻辑结构
- 逻辑结构设计的步骤
  - 将概念结构转化为一般的关系、网状、层次模型
  - 将转换来的关系、网状、层次模型向特定DBMS支持下的数据模型转换
  - 对数据模型进行优化

逻辑结构设计(续)



逻辑结构设计时的3个步骤

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.1 E-R图向关系模型的转换

- 转换内容
- 转换原则

E-R图向关系模型的转换（续）

- E-R图向关系模型的转换要解决的问题
  - 如何将实体型和实体间的联系转换为关系模式
  - 如何确定这些关系模式的属性和码
- 转换内容
  - 将E-R图转换为关系模型：将实体、实体的属性和实体之间的联系转换为关系模式。

E-R图向关系模型的转换（续）

实体型间的联系有以下不同情况：

(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
- 与某一端实体对应的关系模式合并

(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
- 与n端对应的关系模式合并

E-R图向关系模型的转换（续）

(3) 一个m:n联系转换为一个关系模式。

例，“选修”联系是一个m:n联系，可以将它转换为如下关系模式，其中学号与课程号为关系的组合码：

选修（学号，课程号，成绩）

E-R图向关系模型的转换（续）

(4)三个或三个以上实体间的一个多元联系转换为一个关系模式。

例，“讲授”联系是一个三元联系，可以将它转换为如下关系模式，其中课程号、职工号和书号为关系的组合码：

讲授（课程号，职工号，书号）

E-R图向关系模型的转换（续）

(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

E-R图向关系模型的转换（续）

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。

E-R图向关系模型的转换（续）

[例] 把图7.30中虚线上部的E-R图转换为关系模型

- 部门实体对应的关系模式

  部门（部门号，部门名，经理的职工号，…）

  - 此关系模式已包含了联系“领导”所对应的关系模式
  - 经理的职工号是关系的候选码

- 职工实体对应的关系模式

  职工（职工号、部门号，职工名，职务，…）

  - 该关系模式已包含了联系“属于”所对应的关系模式

E-R图向关系模型的转换（续）

[例] 把图7.30中虚线上部的E-R图转换为关系模型（续）

- 产品实体对应的关系模式

  产品（产品号，产品名，产品组长的职工号，…）

- 供应商实体对应的关系模式

  供应商（供应商号，姓名，…）

- 零件实体对应的关系模式

  零件（零件号，零件名，…）

E-R图向关系模型的转换（续）

[例] 把图7.30中虚线上部的E-R图转换为关系模型（续）

- 联系“参加”所对应的关系模式

  职工工作（职工号，产品号，工作天数，…）

- 联系“供应”所对应的关系模式

  供应（产品号，供应商号，零件号，供应量）

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.2 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（续）

- 优化数据模型的方法

1. 确定数据依赖

   按需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间数据依赖

2. 消除 冗余的联系

   对于各个关系模式之间的数据依赖进行极小化处理，消除 冗余的联系。

3. 确定所属范式

   - 按照数据依赖的理论对关系模式逐一进行分析
   - 考查是否存在部分函数依赖、传递函数依赖、多值依赖等
   - 确定各关系模式分别属于第几范式

数据模型的优化（续）

4. 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，

确定是否要对它们进行合并或分解。

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（续）

例：在关系模式

学生成绩单(学号,英语,数学,语文,平均成绩)

中存在下列函数依赖：

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

数据模型的优化（续）

显然有：

学号→(英语,数学,语文)

因此该关系模式中存在传递函数信赖，是2NF关系

虽然平均成绩可以由其他属性推算出来，但如果应用中需要经常查询学生的平均成绩，为提高效率，仍然可保留该冗余数据，对关系模式不再做进一步分解

数据模型的优化（续）

5. 按照需求分析阶段得到的各种应用对数据处理的要求，对关系模式进行必要的分解，以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

数据模型的优化（续）

- 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

数据模型的优化（续）

- 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

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.3 设计用户子模式

- 定义用户外模式时应该注重的问题

  包括三个方面：

  (1) 使用更符合用户习惯的别名

  (2) 针对不同级别的用户定义不同的View，以满足系统对安全性的要求。

  (3) 简化用户对系统的使用

设计用户子模式（续）

[例] 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），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）

- 顾客视图中只包含允许顾客查询的属性
- 销售部门视图中只包含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
- 可以防止用户非法访问不允许他们查询的数据，保证系统的安全性

逻辑结构设计小结

- 任务
  - 将概念结构转化为具体的数据模型
- 逻辑结构设计的步骤
  - 将概念结构转化为一般的关系、网状、层次模型
  - 将转化来的关系、网状、层次模型向特定DBMS支持下的数据模型转换
  - 对数据模型进行优化
  - 设计用户子模式

逻辑结构设计小结

- E-R图向关系模型的转换内容
- E-R图向关系模型的转换原则

逻辑结构设计小结

- 优化数据模型的方法

1. 确定数据依赖

2. 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

3. 确定各关系模式分别属于第几范式。

4. 分析对于应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

5. 对关系模式进行必要的分解或合并

逻辑结构设计小结

- 设计用户子模式

1. 使用更符合用户习惯的别名

2. 针对不同级别的用户定义不同的外模式，以满足系统对安全性的要求。

3. 简化用户对系统的使用

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

7.6 数据库的实施和维护

7.7 小结

7.5 数据库的物理设计

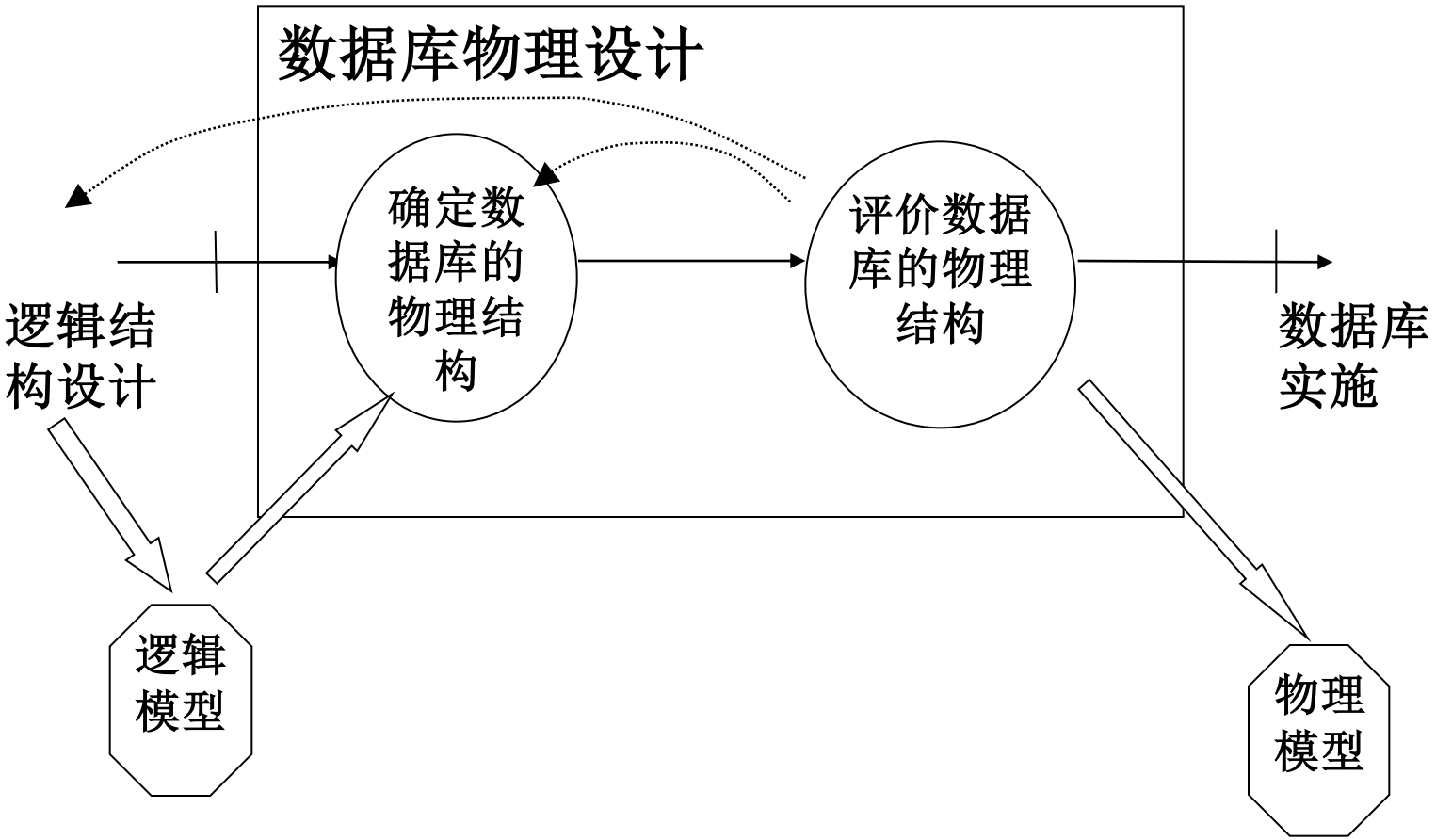
- 数据库的物理设计
  - 数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于选定的数据库管理系统
  - 为一个给定的逻辑数据模型选取一个最适合应用环境的物理结构的过程，就是数据库的物理设计

数据库的物理设计(续)

- 数据库物理设计的步骤
  - 确定数据库的物理结构，在关系数据库中主要指存取方法和存储结构
  - 对物理结构进行评价，评价的重点是时间和空间效率

  如果评价结果满足原设计要求，则可进入到物理实施阶段，否则，就需要重新设计或修改物理结构，有时甚至要返回逻辑设计阶段修改数据模型

数据库的物理设计(续)



7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.1 数据库物理设计的内容和方法

- 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

数据库的物理设计的内容和方法（续）

- 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

数据库的物理设计的内容和方法（续）

- 选择物理数据库设计所需参数(续)
  - 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

数据库的物理设计的内容和方法（续）

- 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

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.2 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
  - 如果一个(或一组)属性经常在查询条件中出现，则考虑在这个(或这组)属性上建立索引(或组合索引)
  - 如果一个属性经常作为最大值和最小值等聚集函数的参数，则考虑在这个属性上建立索引
  - 如果一个(或一组)属性经常在连接操作的连接条件中出现，则考虑在这个(或这组)属性上建立索引
- 关系上定义的索引数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（或属性组）的查询速度，把这个或这些属性（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，可以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
  - 从聚簇中删除经常进行全表扫描的关系；
  - 从聚簇中删除更新操作远多于连接操作的关系；
  - 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，可以选择HASH存取方法
    - 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

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.3 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：

- 数据库数据备份、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，可以考虑存放在磁带上
- 如果计算机有多个磁盘或磁盘阵列，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

确定数据的存放位置（续）

例（续）：

- 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
  - ……

7.5 数据库的物理设计

7.5.1 数据库物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.4 评价物理结构

- 评价内容
  - 对数据库物理设计过程中产生的多种方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构

评价物理结构(续)

- 评价方法（完全依赖于所选用的DBMS ）
  - 定量估算各种方案
    - 存储空间
    - 存取时间
    - 维护代价
  - 对估算结果进行权衡、比较，选择出一个较优的合理的物理结构
  - 如果该结构不符合用户需求，则需要修改设计

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

7.6 数据库的实施和维护

7.7 小结

7.6数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.1 数据的载入和应用程序的调试

- 数据的载入
- 应用程序的编码和调试

数据的载入

- 数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。

- 数据装载方法
  - 人工方法
  - 计算机辅助数据入库

应用程序的编码和调试

- 数据库应用程序的设计应该与数据设计并行进行
- 在组织数据入库的同时还要调试应用程序

7.6数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.2 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
- 如果不满足，对应用程序部分则要修改、调整，直到达到设计要求

2）性能测试

- 测量系统的性能指标，分析是否达到设计目标
- 如果测试的结果与设计目标不符，则要返回物理设计阶段，重新调整物理结构，修改系统参数，某些情况下甚至要返回逻辑设计阶段，修改逻辑结构

数据库的试运行（续）

强调两点：

- 分期分批组织数据入库
  - 重新设计物理结构甚至逻辑结构，会导致数据重新入库。
  - 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。

7.6数据库实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.3 数据库的运行与维护

- 数据库试运行合格后，数据库即可投入正式运行。
- 数据库投入运行标志着开发任务的基本完成和维护工作的开始
- 对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。
  - 应用环境在不断变化
  - 数据库运行过程中物理存储会不断变化

数据库的运行与维护（续）

- 在数据库运行阶段，对数据库经常性的维护工作主要是由DBA完成的，包括：

1. 数据库的转储和恢复
2. 数据库的安全性、完整性控制
3. 数据库性能的监督、分析和改进
4. 数据库的重组织和重构造

数据库的运行与维护（续）

- 数据库的重组织和重构造
  - 重组织的形式
    - 全部重组织
    - 部分重组织
      - 只对频繁增、删的表进行重组织
  - 重组织的目标
    - 提高系统性能

数据库的运行与维护（续）

- 重组织的工作
  - ➢按原设计要求
    - – 重新安排存储位置
    - – 回收垃圾
    - – 减少指针链
  - ➢数据库的重组织不会改变原设计的数据逻辑结构和物理结构

数据库运行与维护（续）

- 数据库重构造

  根据新环境调整数据库的模式和内模式

  - 增加新的数据项
  - 改变数据项的类型
  - 改变数据库的容量
  - 增加或删除索引
  - 修改完整性约束条件

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

7.6 数据库的实施和维护

7.7 小结

7.7 小结

- 数据库的设计过程
  - 需求分析
  - 概念结构设计
  - 逻辑结构设计
  - 物理设计
  - 实施和维护

小结（续）

- 数据库各级模式的形成
  - 数据库的各级模式是在设计过程中逐步形成的
  - 需求分析阶段综合各个用户的应用需求（现实世界的需求）
  - 概念设计阶段形成独立于机器特点、独立于各个DBMS产品的概念模式（信息世界模型），用E-R图来描述

小结（续）

- 在逻辑设计阶段将E-R图转换成具体的数据库产品支持的数据模型如关系模型，形成数据库逻辑模式。然后根据用户处理的要求，安全性的考虑，在基本表的基础上再建立必要的视图（VIEW）形成数据的外模式
- 在物理设计阶段根据DBMS特点和处理的需要，进行物理存储安排，设计索引，形成数据库内模式