**1.概论**

**(1)数据管理的发展阶段及各阶段的特点；**

1人工管理阶段（20世纪50年代）

技术条件

• 外部存储器没有磁盘等直接存取设备。

• 只有汇编语言，无数据管理方面的软件。

• 数据处理的方式基本上是批处理。

• 计算机主要用于科学计算。

数据管理特点

• 数据对程序不具有独立性。

• 数据不共享，程序间有大量的数据重复。

• 数据采用顺序存储方式存储，数据的逻辑结构与物理结构相同。

• 数据不保存在计算机内。

局限

（1） 数据被安排在程序中，数据不具独立性

（2） 同程序的数据之间不共享数据

（3） 数据不保存

2文件系统阶段（20世纪50年代后期至60年代后期）

技术条件

• 外部存储器已有磁盘、磁鼓等直接存取设备。

• 出现了高级语言和操作系统，操作系统中的文件系统用于管理外存数据 。

• 数据处理的方式有批处理和联机实时处理。

• 计算机不仅用于科学计算，还用于信息管理。

数据管理特点

• 出现了专门用于数据管理的软件——文件管理系统。

• 数据以“文件”形式可长期保存。

• 文件组织已多样化，有索引文件、链接文件和直接存取文件等。

• 文件中的数据可顺序访问或随机访问，数据逻辑结构与物理结构已有区别。

• 数据不再属于某个特定的程序，可以重复使用。

• 数据和程序间具有“设备独立性”，即程序只需用文件名就可与数据打交道，由操作系统提供存取方法（读、写）。

• 对数据的操作以记录为单位。文件中只存储数据，不存储文件记录的结构描述信息。文件的建立、存取、查询、插入、修改、删除等所有操作都要由程序来实现。

局限

（1）数据冗余：数据在不同文件中重复存储。

（2）数据独立性较差：文件结构的设计仍然是基于特定的用途，程序基于特定的物理结构和存取方法，因此程序与数据结构之间的依赖关系并未根本改变。

（3）数据联系弱：不同文件中的数据是相互独立的，缺乏相互关联的方法。

3数据库系统阶段（20世纪50年代后期至60年代后期）

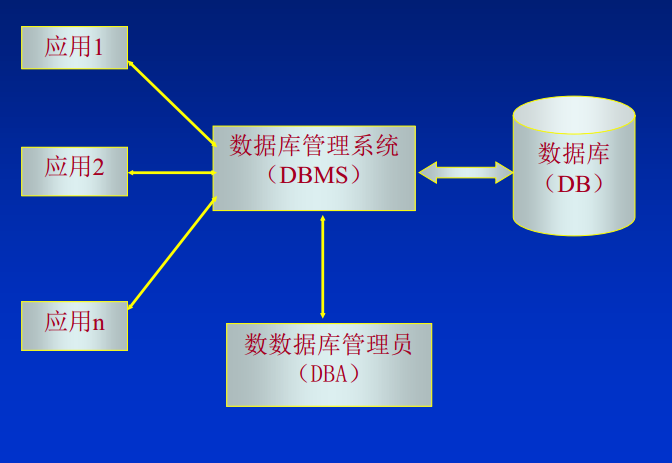
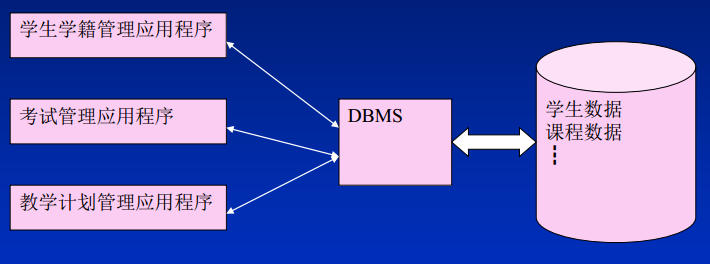
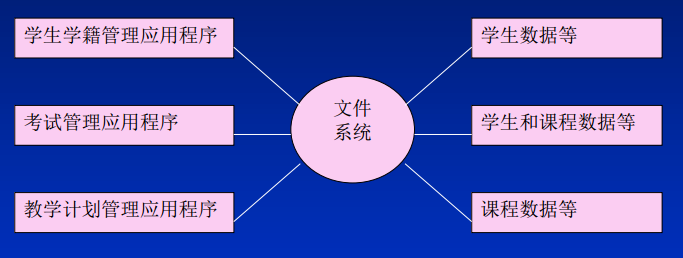
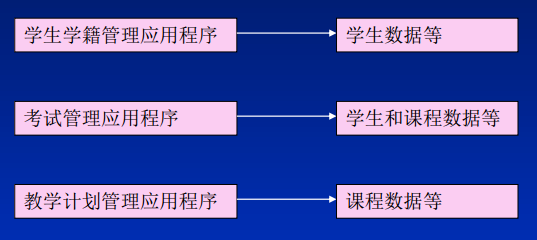
数据管理特点

• 整体数据的结构化。数据库中的数据模型不仅描述数据本身的特征，而且还描述数据之间的联系，且联系通过存取路径（指针）来实现整体数据的结构化。

• 数据的独立性高。数据的存储独立于数据的应用，其逻辑结构不同于物理结构。允许用户以简单的逻辑结构操作数据而无需考虑数据的存储情况。数据的物理位置和存储结构改变时，不必修改和重写应用程序。

• 数据的共享性高、冗余度低，易扩充。允许用户以记录或数据项为单位进行访问，也允许多关键字检索和文件之间的交叉访问。

• 提供了数据安全性、数据完整性、并发访问、数据库恢复等完整的控制功能。



**(2)数据库系统的组成;**

数据库系统（Database System， DBS）是由数据库、软件、硬件和用户四部分组成。

– 数据库：是指长期存储在计算机内、有组织的、统一管理的相关数据的集合。

– 软件：操作系统、数据库管理系统、数据库应用系统等。

– 硬件：存储设备、 CPU、控制器、内存等。

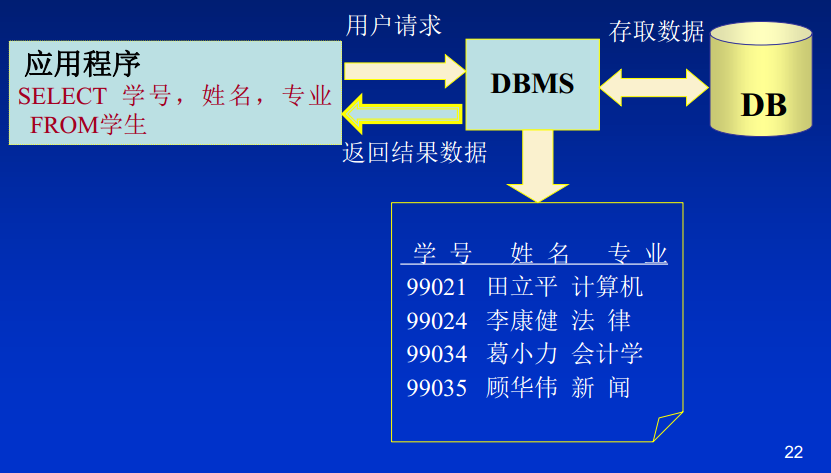
– 用户：数据库管理员、系统分析与设计人员、应用程序员、终端用户

(3)数据库管理员（DBA）的职责；

数据库管理员（Database Administrator，简称DBA），是从事管理和维护数据库管理系统(DBMS)的相关工作人员的统称，属于运维工程师的一个分支，主要负责业务数据库从设计、测试到部署交付的全生命周期管理。

DBA的核心目标是保证数据库管理系统的稳定性、安全性、完整性和高性能。

(4)数据库管理系统（DBMS）组成及功能；



• 一般是由许多“系统程序”所组成的一个集合，不同的DBMS其组成也有所不同，但一般都有以下三部分组成： 1.语言编译处理程序 2.系统运行控制程序 3.系统建立、维护程序

1.数据库的定义功能。DBMS提供模式DDL（描述概念模式的数据定义语言）定义数据库的三级结构、两级映象，定义数据的完整性约束、保密限制等约束。因此，在DBMS中应包括DDL的编译程序。

2.数据库的操纵功能。DBMS提供DML（数据操纵语言）实现对数据的操作。基本的数据操作有两类：检索（查询）和更新（包括插人、删除、更新）。因此，在DBMS中应包括DML的编译程序或解释程序。依照语言的级别，DML又可分成过程性DML和非过程性DML两种。

3.数据库的保护功能。DBMS对数据库的保护主要通过四个方面实现：1、数据库的恢复。在数据库被破坏或数据不正确时，系统有能力把数据库恢复到正确的状态。2、数据库的并发控制。在多个用户同时对同一个数据进行操作时，系统应能加以控制，防止破坏DB中的数据。3、数据完整性控制。保证数据库中数据及语义的正确性和有效性，防止任何对数据造成错误的操作。4、数据安全性控制。防止未经授权的用户存取数据库中的数据，以避免数据的泄露、更改或破坏。

4.数据库的维护功能。这一部分包括数据库的数据载人、转换、转储，数据库的改组以及性能监控等功能。

5.数据字典/系统目录。数据库系统中存放三级结构定义的数据库称为数据字典（DD）。对数据库的操作都要通过DD才能实现。DD中还存放数据库运行时的统计信息，例如记录个数、访问次数等。

**(5)数据模型的概念、组成、类型；**

概念：

模型：是对现实世界中某个对象特征的模拟和抽象。

数据模型：是对现实世界的数据的抽象描述。在数据库技术中，用数据模型描述数据库的结构和语义，即数据模型表示了数据库中数据的组织形式及数据所代表的意义

组成：在数据库领域中，用数据模型描述数据的整体结构，包括三部分：

• 数据结构。描述对象与对象之间的联系，是系统静态的描述。

• 数据操作。DB中对各对象操作的集合，包括操作及有关规则，是系统动态的描述。

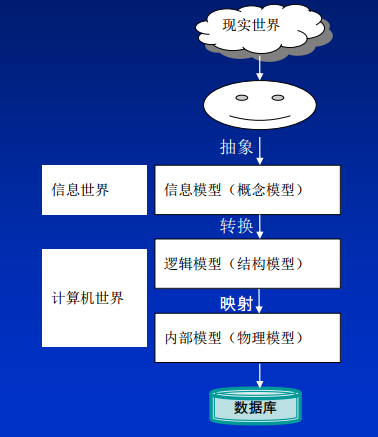
• 完整性约束条件。是一组完整性规则。

类型：按数据模型的抽象层次不同，将其分为3类：

• 概念数据模型，如实体-联系模型

• 结构数据模型（逻辑数据模型）：

包括层次模型、网状模型、关系模型、面向对象模型

• 物理数据模型（逻辑数据模型）

每一种结构数据模型都有其对应的物理数据模型

第一级抽象实现现实世界到信息世界的转换。

信息世界是客观世界在人们头脑中的反映和抽象。现实中的事物被抽象为信息世界中的实体。第一级抽象的结果形成概念数据模型，通常用实体-联系模型表示。

信息世界中的实体经过加工、编码到计算机世界中，则称为数据，即信息数据化。信息世界到计算机世界的变换是第二级抽象（转换）。

**(6)实体、属性、联系、码等基本术语；**

**概念模型：**在概念上表示数据库中将存储哪些信息，而不管这些信息在DB中怎么实现存储。概念模型独立于计算机的硬件、软件和DBMS。最常见的概念模型是实体-联系模型。

1.实体：客观存在的并可相互区别的事物。

实体可以是人，也可以是物，可以是实际的东西，也可以指概念性的东西，还可以指事物与事物之间的联系。

2. 实体集：性质相同的同类实体的集合。

3. 属性：实体所具有的某一特征。其中，型与值分别指名称和数值

4. 实体型：指若干个属性型组成的集合表征一个实体的类型，简称实体型。

实体名(属性名1，属性名2)

5. 键（码）：唯一标识实体的属性或属性集。

6. 联系。实体性之间联系有：一对一、一对多、多对多

**结构模型：**侧重于描述在数据库中数据的存储结构， 即实现数据和数据之间联系的方法。

主要有层次、 网络和关系模型和面向对象模型，分别对应层次、 网络、 关系、 面向对象数据库系统。所以实现模型依赖于DBMS

(7)层次模型、网状模型、**关系模型**的特点；

1.层次模型：可以用倒置树形式表示，实现数据联系的方法是链指针。层次结构的一个重要特点是每一个父可以有多个子，每一个子只能有一个父。层次数据库的父片段与子片段之间是一对多联系，层次结构是按照一对多的原则构造层次树。层次模型的最大缺点是很难表示“一子多父”（多对多联系）的情况。

2.网络模型：网络模型的节点表示存储的记录，用指针实现记录与记录之间的联系。网络模型允许节点无父节点，或有一个以上父节点，从而构成了比层次结构更复杂的网络结构（多对多联系）。例如，学生、课程和成绩这种“一子两父”的结构。这是网络模型与层次模型的本质区别。

3. 关系模型：基本结构是表，表又称为关系。表是由行/列组成的矩阵，表与表之间的联系是通过实体之间的公共属性实现的。

**关系**：一个二维表表示一个关系

**属性**：表中的一列

**域**：属性的取值范围

**元组**：表中的一行

**分量**：一行中的一个属性值

**码**：唯一确定一个元组的属性或属性集

4.面向对象模型：

对象：描述客观事物的一个实体。

对象标识：对象的唯一标识。

封装：把对象内部的属性和操作与对象外部隔离的一种抽象。

类：相同属性和方法的一组对象的集合。

继承：子类继承父类的特征。

**(8)数据库系统DBMS的体系结构：三级结构、二级映射。**

数据库管理系统软件商品多种多样但现在的大型商品数据库管理系统都具有三级模式结构的共同特征。

模式(schema)是数据库中全体数据的特征及其联系的结构化描述，它仅仅涉及到型的描述，不涉及具体的值。模式的一个具体值称为模式的一个实例(instance)。模式是相对稳定的，而实例是相对变动的，因为数据库中的数据是在不断更新的。模式反映的是数据的结构及其联系，而实例反映的是数据库某一时刻的状态。

**三级结构：**

（1）模式：也称逻辑模式，它位于数据库三级结构的中间层，与硬件和软件无关。它是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共视图。

（2）外模式：也称子模式或用户模式，它是数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，是与某一个应用有关的数据的逻辑表示。一个数据库可以有多个外模式。

（3）内模式：也称存储模式，是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。

**二级映射：**

(1)外模式/模式映射：确定外模式和模式之间的对应关系。该映射是在外模式中描述定义的。

 当模式发生改变时（如增加新的关系、新的属性、改变属性和数据类型等），经DBA对外模式/模式映射作相应的改变，就可以使外模式保持不变。

 由于应用程序是依赖于数据的外模式编写的，这样使得模式改变时应用程序不必修改，从而保证了数据与应用程序之间的逻辑独立性，简称数据的逻辑独立性。

(2)模式/内模式映射：确定数据全局逻辑结构与物理结构之间的对应关系。

数据库中只有一个模式，也只有一个内模式，所以模式/内模式映射是唯一的。这个映射是在模式中描述定义的。

当数据库的存储结构发生变化时（如改变了存储位置或采用了另一种存储结构），此时经DBA对模式/内模式映射作相应的改变，就可以使模式保持不变，从而应用程序也不必改变。这就保证了数据和应用程序之间的物理独立性，简称数据的物理独立性。

**三级模式的优点**

（1）保证数据的独立性。数据的独立性分为数据的物理独立性和数据的逻辑独立性。三级模式结构将外模式与逻辑模式分开，保证了数据的逻辑独立性；将逻辑模式与内模式分开，保证了数据的物理独立性。

（2）简化了用户接口，方便了用户使用。用户只要按照外模式编写应用程序或在终端上敲入操作命令，不必了解数据的存储结构。

（3）有利于数据共享。根据不同的业务需求，对应于同一模式可产生不同的外模式，减少了数据冗余，有利于数据共享。

（4）有利于数据的安全保密。用户程序只操作给定的某外模式数据，而不能操作其他数据，这样既保证了数据的安全，也缩小了程序错误传播的范围。

**2.关系模型与关系代数**

**(1)基本概念：域、笛卡尔积、超码、候选码、主码、外码、主属性和非主属性；**

**关系**是行与列交叉的二维表。

关系中的一个**元组**对应表中的一行，

关系中的一个**属性**对应表中的一列，

每一个属性有一个取值范围，称为**属性域**。

**域**是一组具有相同数据类型的值的集合。关系中域表示属性的取值范围，域中包含的值的个数为域的基数(通常用m表示)。

域的表示方式：域名={值1， 值2…， 值N} m=N

**笛卡尔积**：给定一组域D1,D2,D3,…,Dn这n个域的笛卡尔积为：

D1xD2xD3x… x Dn={d1,d2,d3,…,dn}其中，每一个元素(d1,d2,d3,…,dn)称为一个元组。元组中的每一个值称为一个分量，它来自相应的域。

**码（Key）**

超码：能够唯一标识一个元组的属性或属性组，称为超码。

侯选码：侯选码是最小超码，它的任意真子集都不能成为超码。

主码：在一个关系中可能有多个侯选码，选取一个侯选码作为主码来唯一标识每一个元组。

外码：如果一个关系R2的一个属性或属性组X不是R2的主码，而是另一个关系R1的主码，则该属性或属性组X称为关系R2的外码。R1为参照关系，R2为被参照关系。

**主属性和非主属性**：包含在任何一个候选码中的属性成为主属性，不包含在任何一个候选码中的属性成为非主属性。

**(2)关系的数学定义及关系的性质；**

**关系**定义：笛卡尔积D1xD2xD3x…xDn的任意一个子集称为定义在域D1,D2,D3,…,Dn上的n元关系，表示为R(D1,D2,D3,…,Dn)其中：R为关系名，n为关系的目或度。

关系中的每个元素是关系中的元组。当n=1,2分别成为一元关系，二元关系。

关系由**关系头和关系体**组成。关系头由属性A1,A2,A3,…,An的集合构成，每个属性对应一个域Di(i=1,2,3,…,n)；关系体由随时间变化的n个元组的集合构成，每个元组由一组属性值的集合构成。关系头是关系的数据结构描述，关系体是指关系结构中的内容或数值。

关系的性质：

（1）关系中没有重复元组

（2）元组的顺序是无关紧要的，即元组的次序可以任意交换

（3）属性的顺序是无关紧要的，即属性的次序可以任意交换

（4）同一个属性名下的属性值是同一类型的数据

（5）不同的属性可来自同一个域

（6）所有的属性值都是原子的，即每一个属性值都必须是不可分的数据项

**(3)关系与关系模式的区别；**

**关系模式**：一个关系的属性表R(A1,A2,A3,…,An)称为关系模式。

**关系数据库模式**：一组关系模式的集合构成关系数据库模式，是对关系数据库结构的描述

**关系数据库**是由一组关系头的集合及其关系体的集合构成。关系头的集合是对关系数据库结构的描述，是关系数据库模式；关系体的集合是关系数据库的内容。

**(4)关系的完整性约束；**

1 . **域完整性**：是指关系中属性的值应该是给定域中的值。

2 . **实体完整性**：关系都有一个主码，关系的主码不能取空值或部分取空值，也不能重复。

3. **参照完整性**：是指关系的外码必须是被参照关系的主码的有效值，或者是“空值”。

4.**用户定义完整性**：用户定义完整性是针对某一具体关系数据库的约束条件。约束条件由应用环境或需求所决定，它反映某一具体应用所涉及的数据必须满足的语义要求。例如某个属性的取值必须惟一、某个非主属性不能取空值、某个属性的取值范围等

**(5)关系代数概念**

关系模型的组成部分之一是**关系操作**。常用的关系操作有查询、更新（插入、删除和修改）

关系操作的特点是**集合操作方式**，即操作的对象和结果都是集合（或关系）。

**关系运算**是设计关系数据语言的基础，关系运算可分为关系代数和关系演算两大类。

1.**关系代数**：基于关系代数的操作语言，称为关系代数语言，简称关系代数。它是用对关系的运算来表达查询要求的。关系代数语言由IBM在一个实验性系统上实现的。

2.**关系演算**：关系演算是基于关系演算的操作语言，称为关系演算语言，简称关系演算。它用谓词演算表达式来表达对关系操作的要求和条件。

（1）**元组关系演算**：元组关系演算是以元组演算为基础的语言，它用元组演算表达式来表达查询结果应满足的要求或条件。 Eg: ALPHA语言。

（2）**域关系演算**：域关系演算是以域演算为基础的语言，它用域演算表达式来表达查询结果应满足的要求或条件。域演算语言的典型代表是QBE。

3.介于关系代数和关系演算之间的**结构化查询语言**SQL（Structured Query Language）。 SQL不仅具有丰富的查询功能，而且还具有数据定义和数据控制功能。 SQL体现了关系数据语言的特点和优点，是关系数据库的标准语言。

关系代数、元组关系演算和域关系演算三种语言在表达能力上是完全相同的，它们和SQL的共同特点是，语言具有完备的表达能力，是非过程化的语言。

**(6)传统的集合运算和专门的关系运算；**

关系代数包括传统的集合运算和专门的关系运算。

**传统集合运算**是把关系看成是元组的集合，对关系进行并、交、差和广义笛卡尔积等运算。

**专门的关系运算**是指对关系进行选择、投影、连接和除法等运算。

|  |  |
| --- | --- |
| 运算类型运算符 | 运算类型运算符 |
| 集合运算 | ∪（并）、 ∩（交）、 -（差）和 ×（广义笛卡尔积） |
| 专门的关系运算 | σ（选择）、 ∏（投影）、 ⋈（连接）和 ÷（除） |
| 算术比较运算 | ＝、 ≠、＞、 ≥、＜和≤ |
| 逻辑运算 | ∧（与）、 ∨（或）和 ¬（非） |

**1.传统的集合运算**：传统的集合运算是二目运算，即两个关系的集合运算。传统的集合运算包括**并、交、差和笛卡尔积**。

除笛卡尔积外，其他运算都要求参加运算的两个关系必须有相同的度（属性个数相同），且相对应的属性值必须取自同一个域。

**R∪S={t | t∈R ∨ t∈S} R∩S={t | t∈R ∧ t∈S} R-S={t | t∈R ∧ t∉S}**

关系的笛卡儿积称为广义笛卡儿积，它是二目运算，但对参加运算的关系没有特殊要求，即任意两个关系都可以进行广义笛卡儿积运算。

设关系R有m个属性，关系S有n个属性，则R和S广义笛卡儿积（记为R×S）是一个具有（m + n）个属性的关系，且R×S元组的前m列是R的一个元组，后n列是S的一个元组。如果R有i个元组，S有j个元组，则R×S有（i×j）个元组。

**R×S={trts | tr∈R ∧ ts∈S}**

其中trts 称为元组连串，它是一个（m+n）的元组，前m个分量为R中的一个m元组，后n个分量为S中的一个n元组。

**2．专门的关系运算**

(1)**选择:**选择符合条件的元组。**σinyear=’98’ ∧ major=‘法学’(S)**

选择运算是对一个关系进行运算，属于单目运算。选择运算是从关系R中选择某些满足给定条件的元组构成一个新的关系。记作： **σF(t)(S)=F(R)={t | t∈R** **∧ F(t)=True}**

其中F为选择条件，它是一个逻辑表达式，其基本形式为：XiθYi，其中θ表示比较运算符，它可以是＝、 <>、＞、 ≥、＜和≤。Xi、Yi可以是属性名、常量或简单函数。

当有多个选择条件时，可用逻辑运算符∧(与)、∨(或)和¬(非)将多个条件连接起来。

(2)**投影：**取某属性（列）中的所有取值。**∏sno,name,major,inyear(S)**

投影运算也是对一个关系进行运算，属于单目运算。投影运算是从关系R中选择某些属性列构成一个新的关系，并且消除结果中的重复元组。记作：**∏A(i)(S) =A(R)={ t[A] | t∈R}**

其中A是R中属性名，t是R的一个元组，t[A]是元组t在属性A上的诸分量。

投影运算是从关系的垂直方向（列的角度）进行选择运算（取子集）的。

(3)**连接：**取两表格属性两两比较取符合元组组合。**S S.sage⋈> T.tageT**

连接也称θ连接，它是双目运算。连接运算是从两个关系的笛卡儿积中选取满足连接条件的元组。

依据连接条件不同分为：当条件为“=”时，称为等值连接；当条件为“<”时，称为小于连接；当条件为“>”时，称为大于连接；

(4)自然连接

自然连接是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组（公共属性），并且在结果中**消除重复的属性列**。

(5)外连接

为了避免信息的丢失，使得在进行自然连接时能保存这些将被舍弃的元组，提出了外连接运算。外连接运算是指在R和S做自然连接时，把**原该舍弃的元组保留在结果关系中**，而在这些元组新增加的属性上填空值（NULL）。（保留带\*一侧，舍去无\*一侧）

外连接有三种形式：左外连接(S\*⋈T)、右外连接(S⋈\*T)和全外连接(S\*⋈\*T)。

(6)除运算

给定关系R(X,Y)和S(Y,Z)，其中X、Y、Z为属性组。R中的Y与S中的Y可以有不同的属性名，但必须来自相同的域集。R与S的除运算得到一个新的关系P(X)，P是R中满足下列条件的元组在X属性列上的投影：元组在X上分量值x的象集Yx包含S在Y上投影的集合。记作：**R÷S={tr[X] | tr∈R∧∏y(S) ⊆Yx}**，其中Yx为x在R中的象集， x= tr[X]。

除运算是同时从行和列角度进行运算。

除法运算是一个二目运算。设关系R和S的度数分别为n和m(设n>m>0)，那么R÷S是一个度为(n-m)的关系。

R÷S是满足下列条件的最大关系： R÷S中的每个元组t与S中每个元组u所组成的元组(t,u)必在关系R中。

为叙述方便，假设S的属性为R中的后m个属性，则R÷S的具体计算过程如下：

1)T= Π 1,2,…,n-m(R)

2)W=(TхS)-R (即计算在T х S中但不在R中的元组)

3)V=Π1,2,…,n-m(W)

4) R÷S=T-V

**(7)关系运算的安全性和等价性**

1. 关系运算的安全性

在关系代数中，基本的操作包括并、差、笛卡尔积、投影和选择，没有集合的补运算，所以关系代数的运算总是安全的。

关系演算则不同，可能出现无穷关系和无穷的验证问题，例如{t|┓(R(t))}表示不属于R的所有元组集合，表示的是一个无限关系。对于(∀u)(R(u))的真假问题，需要对所有的u进行验证，变为一个无限验证问题，因此在关系演算中需要采取措施防止无限关系和无限验证问题的出现。

在数据库技术中，不产生无限关系和无限验证的运算称为安全运算，与其相对应的表达式称为安全表达式，采取的措施称为安全约束。在关系演算中约定，运算只对表达式中公式在涉及到的关系的值范围内操作，这样就不会产生无限关系和无限验证问题，因此关系演算也是安全的。

2. 关系运算的等价性

并、差、笛卡尔积、投影和选择是关系代数的基本操作，构成了关系代数运算的最小完备集。在这个基础上，关系代数、安全的元组演算、安全的域演算在关系的表达和操作上能力上**完全等价**的。

**(9)查询优化方法。**

在关系代数表达式中通常包含有若干关系操作，系统应该按什么样的顺序执行这些操作，才能够执行起来既省时间又省空间，执行的效率比较高呢？

在关系代数的运算中，**连接运算和笛卡尔积最费时间**，主要原因是当R有m个元组， S有n个元组时，R和S的笛卡尔积就有m×n个元组。当R和S都比较大时，R和S占用的空间比较大，不能够同时读入内存，只能将R和S的部分元组读入内存，重复读写外存，造成执行效率低下，如何花费较少的时间和空间完成R和S的笛卡尔积的运算，就需

要有一个策略问题。

两个关系代数表达式等价是指用同样的关系实例代替两个关系代数表达式时所得到的结果相同。但得到的元组的属性的顺序可以不同，可以记为E1≡E2。

（1）连接和笛卡尔积的交换律**E1⋈E2=E2⋈E1，E1×E2=E2×E1**

（2）连接和笛卡尔积的结合律**(E1⋈E2)⋈E3=E1⋈(E2⋈E3), (E1×E2)×E3=E1×(E2×E3)**

（3）投影的串接 **∏L1(∏L2…(∏Ln(E)))= ∏L1(R),要求L1 ⊆ L2… ⊆ Ln**

（4）选择的串接 **σF1(σF2(σF3(E))) = σF1∧F2∧F3(E)**

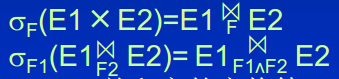
（5）投影和选择的交换 **σF(∏L(E)) = ∏L(σF(E)),** 要求F涉及L中的属性,否则不成立

（6）选择对笛卡尔积的分配律**σF(E1×E2)= σF(E1)×E2,** 要求F只涉及E1中的属性

（7）选择对并的分配律 **σF(E1∪E2)= σF(E1)∪σF (E2)**

（8）选择对自然连接的分配律 **σF(E1⋈E2)= σF(E1)⋈σF (E2)**

（9）选择对差集的分配律**σF(E1-E2)= σF(E1)-σF (E2)**

（10）选择对笛卡尔积的分配律

（11）并和交的交换律E1∪E2= E2∪E1 E1 ∩ E2= E2 ∩ E1

（12）并和交的结合律E1∪(E2∪E3)=(E1∪E2)∪E3 E1∩(E2∩E3)=(E1∩E2)∩E3

此处介绍的优化策略与存储技术无关，主要是如何安排操作顺序。我们只讨论优化的一般技术。优化的策略主要考虑以下几点：

（1）在关系代数表达式中**尽可能早地处理选择运算**。原因是可以尽量减少元组个数，从而得到比较小的关系，减少运算量和读外存的次数。

（2）**尽可能把笛卡尔积和随后的选择运算合并为F连接运算**。原因是减少合并的元组数，减少时间和空间的开销。

（3）**尽可能同时计算一连串的投影和选择**运算。原因是可以避免多次扫描文件，节省操作时间。

（4）表达式中重复出现同一个子表达式时，可将该子表达式的结果计算出来后把结果保存。原因是可以**避免重复计算**。

（5）适当对关系文件做**预处理**。如建立**索引或排序**，可以提高关系的关联速度。

（6）计算表达式之前应先预估顺序的执行顺序。

关系代数表达式的优化是由DBMS的DML编译器完成的。

对一个关系代数表达式进行语法分析，获得语法树，语法树的叶子是关系，非叶结点是关系代数的操作，利用等价变换规则和优化的策略实现对关系代数表达式的优化。算法：关系代数表达式优化

输入：一个关系代数表达式的语法树

输出：计算表达式的一个优化程序

优化方法是依次执行下列的每一步骤。

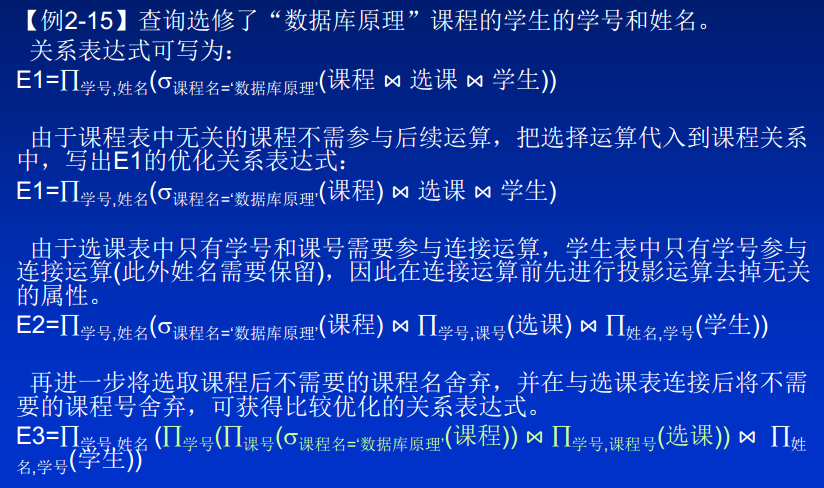
（1）将形如**σF1∧F2∧...∧Fn(E)**的子表达式转换为串接形式：**σF1(σF2(…σFn(E)))**

（2）**尽可能早地执行选择操作**。即把选择操作尽可能移近叶子。

（3）**尽可能早地执行投影操作**。即把投影操作尽可能移近叶子。如果投影是针对表达式的全部属性，则取消该投影操作。

（4）把选择和投影合并为单个的选择和投影或一个选择后跟一个投影，以便使得多个选择、投影能同时执行或在一次扫描中完成。

（5）生成一个程序，每一组结点计算是程序中的一步，各步的顺序是任意的，只要保证任何一组不会在它的子孙组之前计算。



**2.关系模型与关系代数**

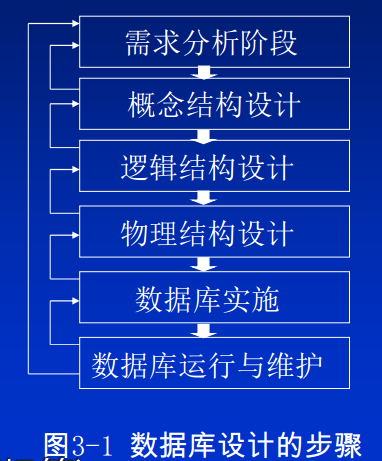
**(1)数据库设计各阶段应完成的任务；**

(2)数据库设计过程中的各级模式；

(3)数据流程图和数据字典在需求分析时的作用；

(4) E-R模型的概念及基本组成；

(5) E-R图与关系模式的转化规则；

(6) E-R模型的设计；

(7) SQL支持关系数据库的三级模式结构；

(8)创建数据库、表和视图的创建、修改和删除。

(9)数据库物理设计的步骤；

(10)数据库的存储和文件结构；

(11)索引与散列。

**(1)数据库设计各阶段应完成的任务**

按照规范设计的方法，可以将数据库设计分为**需求分析**、**概念结构设计**、**逻辑结构设计**、**物理结构设计**、**数据库实施**和**数据库运行**与维护6个阶段。

1、2阶段面向现实世界或用户的应用需求，即面向“问题”，与DBMS无关；

3、4阶段与DBMS密切相关，即面向DBMS；

5、6阶段面向“实现”。

**(2)数据库设计过程中的各级模式；**

①在需求分析阶段，综合各个用户的应用需求；

②在概念结构设计阶段，则产生**独立于计算机硬件和DBMS的概念模式**，它从用户需求的观点描述了数据库的全局概念结构；

③在逻辑结构设计阶段，产生**依赖于DBMS的逻辑模式**，它是由概念模式转换而来，从计算机实现的观点描述了数据库的全局逻辑结构；并根据用户使用数据的要求及安全性，为不同的用户建立不同的视图，即产生外模式，它是从用户使用的观点描述数据库的局部逻辑结构；

④在物理结构设计阶段，根据DBMS特点和处理需要，进行**物理存储设计，产生内模式**在概念结构设计阶段，设计人员从用户需求的观点出发对数据进行建模，产生一个独立于计算机硬件和DBMS的概念模型。

**(3)数据流程图和数据字典在需求分析时的作用；**

**需求**是用户要求数据库应用系统必须满足的所有功能和限制，它包括：功能要求、性能要求、可靠性要求、安全性和完整性要求及开发费用、 开发周期等限制。

- 其中功能要求又包括信息要求和处理要求。

**需求分析**就是通过与用户的沟通和交流获取用户的需求，并对需求进行分析和整理，最终形成需求文档。

**需求获取**：通过用户访谈、开调查会、问卷调查和参加业务实践等方法获取用户实际需求。

（1）调查组织结构：要建立数据库应用系统，首先要清楚当前系统的组织结构情况，即了解该组织各部门的划分及其相互关系、各部门的职责、人员配备、业务分工等。调查结果可用**组织结构图**来描述。

（2）调查管理功能：该功能指的是完成某项工作的能力。每个系统都有一个总目标，为了达到总目标，必须完成各个子系统的功能，子系统的功能又依赖于其下面各项更具体功能的实现。在调查中，可以用**功能层次图**来描述从系统目标到各项功能的层次关系。

（3）调查各部门的业务流程：调查各部门的处理业务、信息来源、处理方法、计算方法、信息流经去向， 提供信息的时间和形态（报告、单据、屏幕显示等）以及安全性和完整性要求。调查结果可用**业务流程图**来描述。

（4）确定新系统的边界：一个组织业务活动的管理不可能全部由计算机来完成，所以设计人员通过对上述调查结果的分析来确定系统的边界，即确定哪些功能由计算机完成或将来让计算机完成，哪些活动由人工完成。由计算机完成的功能就是新系统要实现的功能。

**需求分析和处理**

**分析方法：**

（1）结构分析方法：是一种面向过程的方法。它以过程为中心建立需求模型，常用的分析工具有数据流程图、数据字典、判断树和判断表等。

（2）面向对象分析方法是运用面向对象的方法，找出描述问题域和系统责任所需的类和对象，定义这些类和对象的属性和服务，以及它们之间所形成的结构、静态联系和动态联系，并产生面向对象的模型。常用的分析工具有用例图、类图、顺序图和状态图。

**分析内容：**

**（1）分析业务流程**

了解某项业务的具体处理过程，发现和处理系统调查工作中的错误和疏漏，修改和删除原系统的不合理部分，在新系统基础上优化业务处理流程。

**（2）分析系统数据**

在调查的基础上，进一步收集和分析数据，主要工作包括：

a.明确用户在数据库中需要存储哪些数据，即确定各实体集以及各实体集所包含的属性。

b.明确实体集之间的联系，即确定联系的类型。

c.明确各属性的组成，即属性的名称、类型、长度、值域、使用特点等。

d.明确各数据的来源及不同来源数据是否有矛盾；数据之间的联系；哪些是基本数据；哪些是导出数据；数据的重要性；及数据可能发生的变化等。

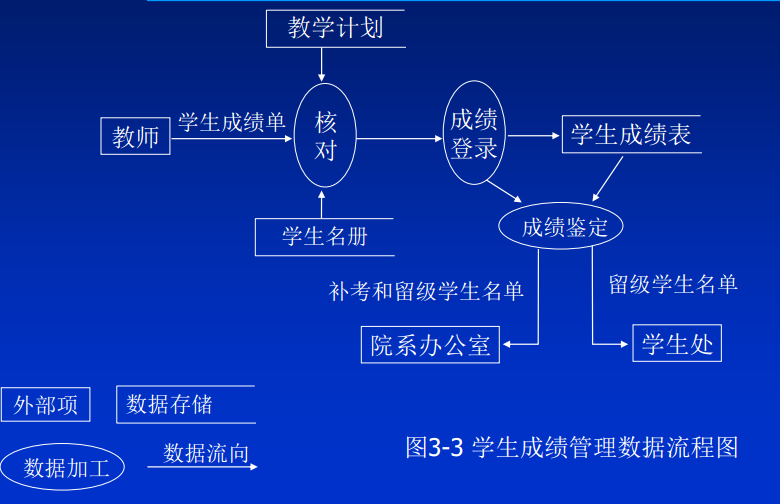
e.明确用户对数据处理的要求，即要完成的具体处理内容、响应时间、安全性和完整性等要求。

**数据流程图(Data Flow Diagram，DFD)**：DFD是一种能全面地描述信息系统逻辑模型的主要工具，它可以用少数几种符号综合地反映出信息在系统中的流动、处理和存储情况。 DFD不涉及技术细节，所描述的内容是面向用户的，它是分析人员与用户进行交流的有效手段。

DFD是分层次的，采取自顶向下逐层分解的方法绘制，具体层数视实际情况而定。

DFD的基本符号有4种：外部项、数据存储、处理过程、数据流向。

数据流程图以图形的方式描述了新系统的逻辑模型框架，对每个图形符号不能给出详细的定义。



通过表格或文字对数据流程图中的每个成分都给出详细的描述和定义，这些描述和定义所组成的集合就是**数据字典**。

数据流程图与数据字典配合，可从图形和文字两方面对系统的信息模型进行完整的描述。

数据字典主要是对数据流程图中的数据项、数据结构、数据流、处理逻辑、数据存储、外部实体6个元素进行具体的定义。

1. **数据项**：数据项又称数据元素，是数据的最小单位，描述数据的静态特性。其定义包含如下内容：数据项的描述={数据项编号，数据项名称，别名，简述，类型及宽度，取值范围，取值含义，存储处} 例如：“学号”数据项的定义如下：

数据项编号： I02—01

数据项名称：学号

别名：学生编号

简述：学号是学生的标识符，每个学生都有惟一的学号。

类型及宽度：字符型， 9位

取值范围： “000000001” ~“999999999”

取值含义：前4位代表学生入学年份，后3位是序号，中间2位系编号。

存储处：学生名册和学生成绩

1. **数据结构**：数据结构描述某些数据项之间的关系。一个数据结构可以由若干个数据项组成；也可以由若干个数据结构组成；还可以由若干个数据项和数据结构组成。其定义包含如下内容：数据结构的描述={数据结构编号，数据结构名称，简述，数据结构组成} 例如：“成绩”数据结构的定义如下：

数据结构编号 : DS01-12

数据结构名称 : 学生成绩单

简述 : 教师所填学生的成绩单

数据结构组成 : 姓名 + 课程名 + 分数

1. **数据流**：数据流由一个或一组固定的数据项组成。其定义内容如下：

数据流的描述={数据流编号，数据流名称，简述，数据流来源，数据流去向，数据流组成，平均数据流量，高峰流量} ：“留级学生名单”数据流的定义如下：

数据流编号： D3

数据流名称：留级学生名单

简述：列出所有留级学生名单

数据流来源：成绩鉴定模块

数据流去向：学生处

数据流组成：学号+姓名+院系+班级+原因

平均数据流量： 10份/学年

高峰流量： 30份/学年

1. **数据存储**：数据存储在数据字典中只描述数据的逻辑存储结构，而不涉及它的物理

组织。其内容如下：数据存储的描述={数据存储编号，数据存储名称，简述，数据存储组成，关键字，相关联的处理} ：“学生成绩表”数据存储的定义如下：

数据存储编号： F3

数据存储名称：学生成绩表

简述：存放学生所选课程的考试成绩

数据存储组成：学号+课程号+分数

关键字：学号+课程号

相关联的处理： P2、 P3

1. **处理逻辑**：处理逻辑的定义仅对数据流程图中最底层的处理逻辑加以说明。内容为：处理逻辑的描述={处理逻辑编号，处理逻辑名称，简述，输入的数据流，处理，输出的数据流，处理频率} ：“核对”处理逻辑的定义如下：

处理逻辑编号： P1

处理逻辑名称：核对

简述：对教师提交的成绩单按有关约定进行核对

输入的数据流：学号和姓名，来源于数据存储“学生名册表”；课程号和课程名，来源于数据存储“教学计划”；分数来源于外部实体“教师”。

处理：根据数据流“学号”和“姓名”，检索学生名册，确定该学生是本校学生；再根据数据流“课程号”和“课程名”，检索数据存储教学计划，以确定该学生的选课；再根据教师输入的分数，得到正确的成绩单。

输出的数据流：数据流“正确的成绩单”去下一个处理逻辑“成绩登录”。

处理频率：每学期处理一次。

1. **外部实体**：外部实体的描述={外部实体编号，外部实体名称，简述，输入的数据流，输出的数据流} 例如：“教师”外部实体的定义如下：

外部实体编号： S1

外部实体名称：教师

简述：任课教师

输入的数据流：无

输出的数据流： D1

**编写需求规格说明书**

在调查分析的基础上，编写需求规格说明书，内容包括：

• 系统的概况、目标、范围、背景、历史和现状；

• 系统总体结构与子系统结构说明；

• 系统功能说明；

• 系统的软硬件支持环境的选择及规格要求；

• 组织结构图、组织之间联系图等；

• 数据流程图、功能模块图和数据字典等。

**(4) E-R模型的概念及基本组成；**

在概念结构设计阶段， 设计人员从用户需求的观点出发对数据进行建模， 产生一个独立于计算机硬件和DBMS的概念模型。

**概念模型**是现实世界到信息世界的第一级抽象，也是设计人员与用户交流的工具之一，因此要求概念模型简单、清晰、易于理解，同时还应具备较强的语义表达能力，可以直接表达用户的各种需求，并易于向数据模型的转换。概念模型的表示方法有很多，目前常用**实体-联系方法**（Entity Relationship Approach） 来表示概念模型。

实体-联系方法是用实体-联系图(Entity Relationship Diagram，E-R图)来描述现实世界的概念模型，它是1976年由P．P．Chen提出的一种常用的概念模型设计工具。

**用E-R方法建立的概念模型也称为E-R模型。E-R图**是用图形化的方法直观地描述概念模型，**其基本组成是实体、属性和联系**，用线段（无向边）将实体与属性、实体与联系或联系与属性连接起来，并在线段旁标注联系的类型，由此而形成E-R图。

**实体（或实体集）用矩形**表示，在数据库中实体通常指某类事物的集合，即实体集；

**属性用椭圆**表示，是对实体特征的描述；**联系用菱形**表示；

A属性

（1）**简单属性和复合属性**。简单属性是指不能划分为更小部分的属性，例如性别。复合属性是指可以划分为更小部分的属性，例如地址。

（2）**单值属性和多值属性**。单值属性是指对一个特定实体只能有一个值的属性，例如学号。多值属性是指对一个特定实体可能对应一组值的属性 ，例如联系电话。

（3）**存储属性和派生属性**。派生属性是指属性的值可以从其他的相关属性或实体派生出来，例如某类商品的销售额，就可通过销售量\*单价得到，其中销售量属性和单价属性称为基

属性或存储属性，派生属性的值不存储，需要时可根据某种算法推导出来

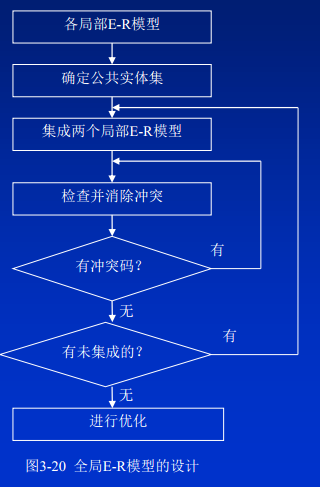
B联系：联系的元数是指参与联系的实体集的数目。如：一元联系、二元联系、N元联系

**(5) E-R模型的设计；**

1.**概念结构设计方法**有四种：

（1）自顶向下方法。首先定义全局概念结构的框架，然后逐步细化，形成各局部应用的概念结构。（2）自底向上方法。首先定义各局部应用的概念结构，然后将它们集成，得到全局概念结构。（3）由里向外方法。首先定义最重要的核心概念结构，然后向外扩张，生成其他概念结构，直至得到全局概念结构。（4）混合方法。将自顶向下和自底向上相结合的方法，用自顶向下方法设计一个全局概念结构的框架，以它为骨架集成由自底向上方法中设计的各局部应用的概念结构。

2.概念结构**设计步骤**：概念结构设计是在需求分析的基础上，设计出系统的概念模型，而概念模型一般用E-R模型表示，所以概念结构设计就是设计E-R模型。

在此采用自底向上方法设计E-R模型，具体步骤是：

• 首先设计各局部应用的E-R模型，

• 然后将局部E-R模型集成为全局E-R模型，

• 最后对全局E-R模型进行优化。

①局部应用的E-R设计

1.确定局部应用

2.确定实体及其属性

• 首先将可以独立区分的事物识别为实体；

• 实体具有一个以上属性；

• 属性若需要进一步描述其特征，应调整为实体。

3.确定实体之间的联系

4.画出局部E-R图

②全局E-R模型设计

1.确定公共实体

2.集成两个局部E-R模型

3.消除冲突

(1)属性冲突：①属性域冲突。②属性取值单位冲突。

(2)结构冲突

①同一事物在不同的局部ER模型中具有不同的抽象，分别为实体和属性。

②同一实体在不同的应用中其属性个数和属性排列次序不同。

③同一联系，在不同的应用中呈现不同的类型，分别为1:1、 1:2或1:n。

(3)命名冲突

①同名异义，不同意义的事物（实体、属性或联系）在不同应用中具有相同的名称。

②异名同义，同一意义的事物（实体、属性或联系）在不同应用中具有不同的名称。

③优化全局E-R模型

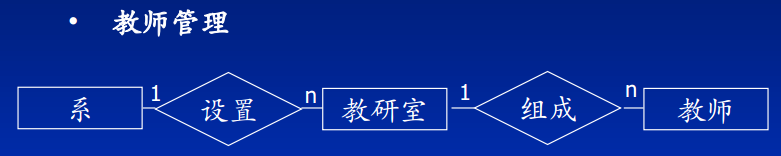
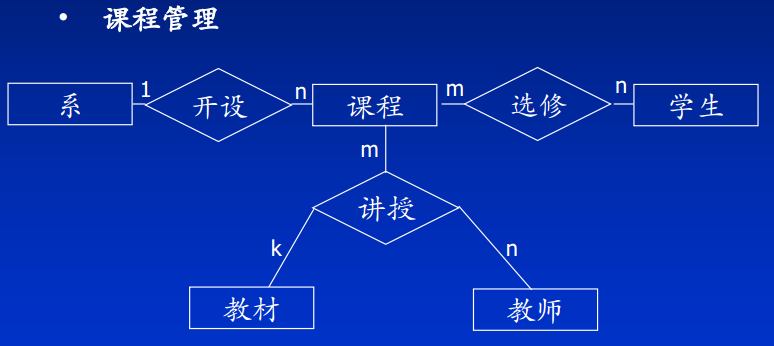
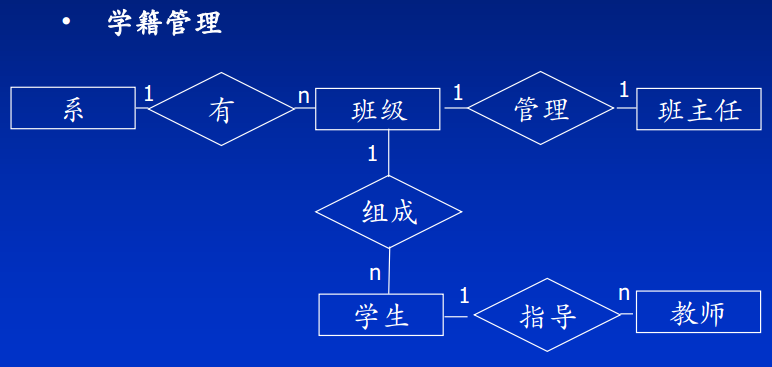
• **优化全局E-R模型**就是对全局E-R图进行修改、重构和优化，得到最佳的全局E-R图方案，使其既能满足所有用户的需求又能减少冗余属性和冗余联系。

• **冗余属性**是指可以由其他属性值推导出来的属性，冗余联系是指可以由其他联系推导出来的联系。冗余属性和冗余联系容易破坏数据库的完整性，增加数据库维护难度。

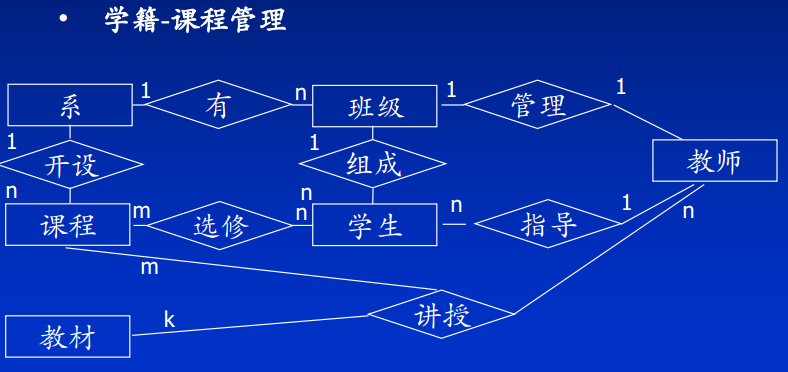
• 减少冗余属性和冗余联系可采用**分析方法**，即以需求阶段所建立的数据字典和数据流程图为依据，根据其中有关数据项之间的逻辑关系确定冗余数据项，从而消除不必要的冗余属性和冗余联系，但有时为了提高效率需要保留必要的冗余。另外，还可用规范化理论消除冗余。

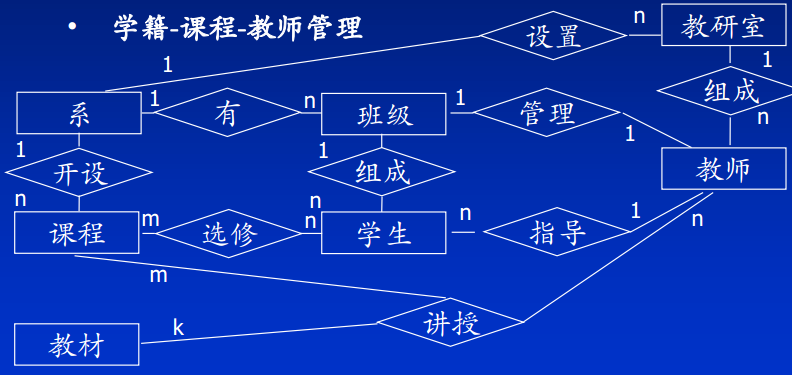
案例：教学管理系统

1.划分为3个局部应用分别设计：学籍管理、课程管理、教师管理



2.依次合并局部模型





(6) E-R图与关系模式的转化规则；

**逻辑结构设计**就是将概念结构设计阶段完成的E-R模型转换为所选择的DBMS所支持的数据模型，并对数据模型进行优化。逻辑结构设计步骤为：

（1）将概念模型转换为一般的关系、层次、网状模型或面向对象数据模型；

（2）将一般的关系、层次和网状模型转换为特定DBMS所支持的数据模型；

（3）优化数据模型。

E-R模型向关系模型的转换主要解决两个问题：

• 一个是如何将实体型和实体间的联系转换为关系模式？

• 另一个是如何确定这些关系模式的属性组成和码？

**转换原则**如下：

1．一个实体型转换为一个关系模式。实体的属性即关系的属性，实体的码即关系的码。

2．依据联系的类型将联系转换为属性或关系模式。

（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。

• 若转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，各实体的码组合起来作为该关系的候选码。

• 若与任意一端对应的关系模式合并，则需要在该端的关系模式的属性中加入另一端关系模式的码和联系本身的属性。

（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。

• 若转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，各实体的码组合起来作为该关系的候选码。

• 若与n端对应的关系模式合并，则需要在n端的关系模式中加入1端的实体的码以及联系本身的属性。

（3） 一个m： n联系只能转换为一个独立的关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，该关系的码为各实体码的组合。

（4） 三元联系转换为一个独立的关系模式。 与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，该关系的码为各实体码的组合。

例子：

①1:1联系：将联系与班级一方的关系模式合并，结果如下：

班级（班号，班名， 班主任编号）

班主任（班主任编号，姓名，职称）

②1:n联系：将联系与n端(课程)的关系模式合并，结果如下：

系（系号，系名，系主任名）

课程（课程号，课程名，学时，开课学期， 系号）

③m:n联系：将联系转换为一个单独的关系模式，结果如下：

学生（学号，姓名，性别，出生日期）

课程（课程号，课程名，学时，开课学期）

选修（学号，课程号，成绩）

④三元联系：将联系转换为一个单独的关系模式，结果如下：

课程（课程号，课程名，学时，开课学期）

教师（教师编号，姓名，性别，出生日期，职称）

教材（ISBN，书名，作者，出版社，版次，单价）

担任（教师编号，课程号， ISBN）

**优化与调整关系模式**

• 数据库逻辑结构设计的结果不是惟一的，哪个结果更好，需要按照第4章介绍的关系规范化理论为指导进行结果的选择。

• 有时为了提高数据库应用系统的性能，还应根据应用需要适当地修改、调整和重构关系模式，以得到优化的关系模式。

• 常用的优化方法是根据应用需要对关系模式进行垂直分割和水平分割。

**垂直分解**： 把一个关系模式R的属性分解为若干子集合形成若干子关系模式。如右图所示，关系R被垂直分解为R1,R2,R3以及R-(R1+R2+R3)。

垂直分解的原则：凡是经常在一起使用的属性从R中分解出来形成一个子关系模式。

垂直分解的方法可以采用简单的直观分解，也可以用关系模式分解算法进行分解。需要注意的是，垂直分解必须以不损失关系模式的语义(保持无损连接性和保持函数依赖性)为前提。

垂直分解的好处是可以提高某些事务的效率；不足之处是可能会使得另一些事务不得不执行连接操作从而降低了效率。是否需要垂直分解，取决于分解后R上的所有事务的总效率是否得到了提高。

例：学生（学号,姓名,性别,出生日期,入学时间,入学成绩,毕业中学,班号）

如果经常查询的是学号、姓名、性别、班号4个属性，其他属性使用较少，为了提高查询速度，可将学生关系模式垂直分解为如下两个关系模式：

学生1（学号，姓名，性别，班号）

学生2（学号，出生日期，入学时间，入学成绩，毕业中学）

**水平分解**： 指把一个关系模式R中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。如右图所示，关系R被水平分解为R1,R2,R3以及R-(R1+R2+R3)。

例如，一个关系很大(这里指元组数多)，而实际应用中，经常使用的数据只是一部分(通常只占兀组总数的20%)，此时可以将经常用的这部分数据分解出来，形成一个子关系，这样可以减少查询的数据量。

例：教师（教师编号，姓名，性别，出生日期，职称）

如果经常按职称分类查询教师的情况，则可将教师关系模式水平分解为多个关系模式：

教授名册（教师编号，姓名，性别，出生日期）

副教授名册（教师编号，姓名，性别，出生日期）

讲师名册（教师编号，姓名，性别，出生日期）

助教名册（教师编号，姓名，性别，出生日期）

例：假设有一个产品关系模式，其中包含有出口产品和内销产品两类数据。由于不同的应用关心不同的产品，如一个应用只关心进口产品，而另一个应用只关心内销产品。因此，可将产品关系模式进行水平分解，分解为两个关系模式，一个存放出口产品数据，另一个存放内销产品数据，如下图所示。这样可以减少应用存取的元组数。

**外模式**是用逻辑模型对用户所用到的那部分数据的描述。外模式也称为**子模式或用户模式**，是与应用程序对应的数据库视图，是数据库的一个子集，也是局部逻辑数据模式。不同的用户可有自己的外模式。

外模式的主要作用如下:

(1)提供一定的逻辑数据独立性

(2)更好地适应不同用户对数据的需求

(3)有利于数据保密

【例】 设有一产品关系模式：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)

可以在产品关系模式上为不同的用户建立两个视图。

为一般客户建立视图：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。这样就可以防止用户非法访问本来不允许他们查询的数据，保证了系统的安全性。

(7) SQL支持关系数据库的三级模式结构；

见第5页。

(8)创建数据库、表和视图的创建、修改和删除。

(9)数据库物理设计的步骤；

**数据库的物理结构**主要是指数据库在物理设备上的存储结构和存取方法。数据库物理设计的任务就是利用所选DBMS提供的手段，为设计好的逻辑数据模型选择一个符合应用要求的物理结构。

由于不同的数据库产品所提供的物理环境、存取方法和存储结构有很大差别，能提供给设计人员使用的设计变量、参数范围也很不相同，因此没有通用的物理设计方法可遵循，只能给出一般的设计内容和原则。

本章讨论的数据库物理设计的内容有：存储结构的确定、簇集设计、索引的选择。

**1.确定记录的存储结构**：数据库逻辑设计阶段所讨论的关系和记录仅仅是逻辑形式，而不是其存储形式。数据库的物理设计阶段要考虑如何表示逻辑记录的内容，即记录的存储形式，同时需要进一步考虑一些其他问题。例如数据项编码是否要压缩，不定长数据项如何表示，记录间互相连接的指针如何设置等。另外，传统数据模型通常是以记录为基础，因此了解记录的一些存储技术和数据压缩方法对物理设计是很有利的。

1、数据项的存储技术

在存储记录中，数据项的表示方法一般有四种：定位法、相对法、标号法以及索引法。

一般而言，所有DBMS都支持定长记录，但只有部分DBMS支持可变长记录。应用中可根据实际情况选择定长存储法或某种可变长存储法。

(1)定位法：定位法是一种只能表示定长记录的方法。使用定位法存储记录时系统为每个数据项按其最大可能长度分配定长的字段。数据项从左向右填入，空白部分填以空白字符，见下图。

优点：简单，使用最广。

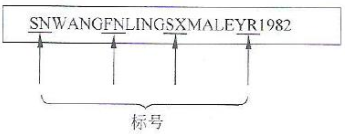
缺点：存储空间利用率不高，尤其是当数据项长度参差不齐时，空间浪费更加突出。

(2)相对法

相对法是一种可表示可变长记录的存储方法。使用相对法存储记录时每个数据项不是定长的，而是由实际情况确定。不同数据项之间用特殊的字符隔开。

相对法与定位法相比，其优点是存储空间的利用率高。

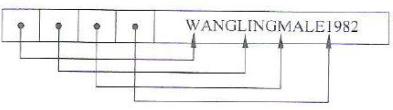
(3)标号法

标号法也是一种可表示可变长记录的存储方法。使用标号法表示存储记录时，每个数据项用一标号开头。

优点：空间利用率高。

缺点：当记录的数据项个数较多时，标号本身也将占用较多的空间

(4)索引法

索引法一也是一种可表示可变长记录的存储方法。使用索引法表示存储记录时每个数据项用一个指针指向其首地址，见下图

索引法的优点也是空间利用率高。

2、记录在物理块上的分配

磁盘与内存交换数据是以物理块为单位进行的，即每次访问磁盘时至少存取一个物理块。在磁盘上，记录必须分配到物理块中。

当记录比物理块小时，一个物理块可容纳多个记录，记录不跨块，这种结构称为不跨块组织。当记录很大，一个物理块容纳不下时，记录必须跨块存放，这种结构称为跨块组织。

一个物理块总有些附加信息，如识别标志、块的类型、指针等，实际用于存储数据的空间要小于块的大小。

变长记录的大小不定，很难确定每块应含有多少个记录，一般也采用跨块组织

3、物理块在磁盘上的分配

在现代的DBMS中，由系统初始化时向操作系统一次性地申请数据库所需要的磁盘空间。物理块在磁盘上的分配方法一般有四种：

(1)连续分配法

这种方法的特点是将一个文件的块分配在磁盘的连续空间上，块的次序也就是它们存储的次序。

优点是对顺序存取多个块或全部文件很有效。缺点是不利于文件的扩充和修改。

(2)连接分配法

这种方法的特点是物理块不一定分配在磁盘上的连续区域，各物理块在磁盘上用指针链接。

优点是有利于文件的扩充和修改。缺点是存取效率低。

(3)簇集分配法

这种方法是上面两种方法的结合，特点是把文件分为若干簇集，即文件中相关记录存放在一个物理块内或相邻的几个物理块内，各簇集以指针链接。

优点是有利于检索速度的提高。缺点是重新分配簇集时会引起大量数据的搬移。

(4)索引分配法

这种方法的特点是每个文件有1个逻辑块号与其物理块地址对照的索引。通过索引，可查询到文件中任一块的地址。

优点:分配方法灵活，扩充、修改较方便。

缺点:访问时增加了逻辑块号到物理块地址映射的开销；由于物理上不一定相邻，故连续访问时性能不高。

4、数据压缩方法

为了节省存储空间，有些DBMS允许对数据进行压缩后存储。但访问这些数据时须进行转换或复原，因此，对系统的性能有影响。通常，数据库中数据压缩技术应用得并不普遍，而且即使应用，也只采用一些简单的数据压缩方法。

(1)消零或空白符法。 如果数据项中常常出现一连串的零或空白符，可以用一种特殊符号及表示零或空白符个数的数字表示。例如，空白符"bbbbb"可用"#5"表示，一连串零"000000"可用"@6"表示。

(2)模式代替法。如果数据项中常常出现一些重复的字符串，此时可以用一省略符代替，并登记在模式表中。

(3)索引法。索引法是模式代替法的变种。

**2.确定数据库的存储结构**，内容包括：确定数据的存放位置、选择文件的组织方式、确定系统配置等。

1、确定数据的存放位置：将不同性质的数据分开存放

通常，影响数据存放位置和存储结构的因素与所选用的硬件环境和应用需求有关，如存取时间、存储空间利用率、维护代价等，而这3个方面常常是相互矛盾的。实际应用时必须进行权衡，选择一种折中方案。确定数据存放位置的一般原则是：

(1)根据实际应用情况，可将经常变动的数据和不经常变动的数据分开存放。

例如，在企业销售管理系统中产品数据不经常变动，而产品的订单数据经常变动，所以订单数据和产品数据可以分开存放。

(2)将访问频率高的数据与访问频率不高的数据分开存放。

例如，数据库的数据备份、日志文件备份等由于只在故障恢复时才使用而且数据最很大，可以考虑存放在磁带或大容最磁盘上。

(3)如果计算机提供多个磁盘，可以考虑将关系表和索引分别存放在不同的磁盘上，查询时，由于两个磁盘驱动器分别工作，因而可以保证较快的物理读写速度。

(4)另外，可将较大的关系表分别存放在两个磁盘上，以加快存取速度，这在多用户环境下特别有效；也可将日志文件与数据库对象(关系表、索引等)放在不同的磁盘以改进系统的性能。

2、选择文件的组织方式：堆、有序文件、 Hash文件、索引顺序文件等

数据库物理设计的主要目标之一就是以有效方式存储数据。例如，如果想按姓名以字母顺序检索职I:记录，则按职下姓名对文件排序就是一种很好的文件组织方式。但是，如果想要检索所有工资在某个范围内的职工，则按职工姓名排序就不是一种好的文件组织方式。因此，如何使用有效的文件组织方式建立数据库是数据库应用中必须考虑的问题。如果所选DBMS提供选择各种文件组织方式的功能，就可以为每个表选择最佳的文件组织方式。常见的文件组织方式有堆文件、有序文件、 Hash文件、索引顺序存取方式以及B+树等。

(1)堆文件：对于向表中插入大量数据的操作，堆文件是最好的文件组织方式。因为记录被顺序地插入到文件的末尾。因此不会在计算记录要插入哪一块时产生额外消耗。缺点：由于是无序文件，查找效率低。被删除的记录的空间是不可再用的，删除量大时空间利用率低。

(2)有序文件：利用二分查找效率高。

(3)Hash文件：只有当检索是基于Hash字段值的准确匹配，尤其是如果访问顺序是随机的时，使用Hash文件才是一种好的存储结钩。不适合于Hash列的模式匹配、值的范围和指定的部分键查询。例如，如果学生表student是基于学号进行Hash映射的，则检索学号为9309203的记录就很有效。

(4)索引顺序存取方式： 这是一种比Hash文件更通用的组织方式。索引顺序存取方式支持基于准确键匹配、模式匹配、值的范围和指定的部分键。但是，由于这种方式的索引是静态的，在创建文件时就要生成。因此，更新表会使索引文件的性能变坏、丢失存取索引键的顺序，使得以索引键的顺序进行检索时变得很慢。不过这个问题可以利用B+树文件组织方式来解决。

(5)B+树：也是一种比Hash文件更通用的组织方式，也支持基于准确键匹配、模式匹配、值的范围和指定的部分键。 B+树的索引是动态的，当表更新时文件的性能不会变坏。但是，如果表并不经常更新，索引顺序存取方法可能比B+树更加有效，因为它少了一级索引， B+树的叶结点包含指向表中实际记录的指针而不是实际记录本身。

3、确定系统配置：确定合适的存储分配参数

通常， DBMS产品都提供了一些存储分配参数，例如同时使用数据库的用户数、同时打开的数据库对象数、使用的缓冲区长度和个数、时间片大小、数据库的大小、装填因子、锁的数目等。

一般情况下，系统都为这此变量赋予了合理的缺省值。但是这此值不一定适合每一种应用环境，在进行物理设计时，需要根据应用环境来确定这些参数值以使系统性能最优。

另外，物理设计时对系统配置变量的调整只是初步的，在系统运行时还要根据系统实际运行情况做进一步的调整，以便改进系统性能。

**3.簇集设计**：所谓**簇集**是指把有关的元组集中在一个物理块内或物理上相邻的若干个物理块内，其目的是提高对某此数据的访问速度。 有些DBMS提供了控制数据存放的功能，如ORACLE就提供了簇集功能，使得用户可根据需要对数据存放位置进行控制。

【例】如果有一个存放职工基本信息的职工文件，应用中经常会根据职工文件查询某某年出生的职工情祝。如查询1973年出生的职工情况。如果1973年出生的职工有50人，最坏情况下，这些职工被存储在50个不同的物理块上。这时如果要查询1973年出生的职工，就要做50次的I/O操作，显然这将影响系统查询的性能。

但是，如果在职工文件上按"出生年月"簇集键集中存放元组，也就是把出生年份相同的职工在物理上存放在一个物理块或相邻的几个物理块上，这样每做一次I/O操作，就可以获得多个满足查询条件的记录，从而显著地减少了访问磁盘的次数。

一般在满足下列**条件**时，才考虑建立簇集:

(1)应用中主要是通过簇集键进行访问或连接的。尤其当主要应用的语句中包含有与簇集键有关的order by, group by, union, distinct等内容时簇格外有利，可省去对结果的排序。

(2)对应每个簇集键值平均元组数要适当。元组数太少，簇集的效益不明显，甚至浪费空间；元组数太多，需采用多个连接块，同样不利于提高性能。

(3)簇集键值应相对稳定，以减少修改簇集键所引起的维护开销。注意：如果应用中发现所设的簇集收效不犬，甚至有害，或者因为应用改变了，这种簇集没有必要了，就应该撤销簇集。

**4.索引的选择**：索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。

如果包含记录的文件根据某个索引键属性顺序排序，那么该索引键对应的索引称为主索引。索引键属性保证每条记录都有唯一的值。通常情况下主索引的索引键是主键，但并非总是如此。

那些其索引键指定的顺序与文件中记录的物理顺序不同的索引称为辅助索引

选择索引存取方法实际上就是根据应用要求确定对关系的哪些属性列建立索引、哪些属性列建立组合索引、哪些索引要设计为唯一索引等。一般来说：

（1）如果一个属性（或属性组）经常在查询条件中出现，则考虑在这个属性（组）上建立索引（或组合索引）；

（2）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；

（3）如果一个属性（或属性组）经常在连接操作的连接条件中出现，则考虑在这个属性（组）上建立索引（或组合索引）。

关系上定义的索引不是越多越好，系统为维护索引要付出代价。

**5.评价物理结构**：评价物理数据库物理结构的方法完全依赖于所选用的DBMS，主要是从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的物理结构。 如果结构不符合用户的需求，则需要修改物理设计。

(10)数据库的存储和文件结构；

(11)索引与散列。

见（9）