数据库课堂笔记

基本概念

DB：存储在计算机内有组织、有结构的数据集合（DBMS和DB没有相互包含的关系，可以用不同的DBMS访问DB），数据库中存储的是数据及**数据之间的联系**

DBMS：数据库管理系统=查询处理器+存储管理器

DBMS的功能：数据定义（DDL）、数据操纵（DML）、数据库的运行控制（实际是管理功能，DCL）、数据库的维护、数据字典（DD，用数据解释数据，元数据），主要功能是定义数据库

DBMS的特点：数据结构化（共享性高）、冗余度低、易扩容、独立性高，数据由DBMS统一控制和管理

DBS的组成：硬件系统、数据库集合、系统软件、数据库管理员、用户（终端用户和程序员）

”数据库“可能指代DB、DBS、DBMS

数据库系统的核心是数据库管理系统，基础是数据模型

数据库中产生数据不一致的根本原因是数据冗余

数据库是系统软件，DBMS调用OS，不是相互调用

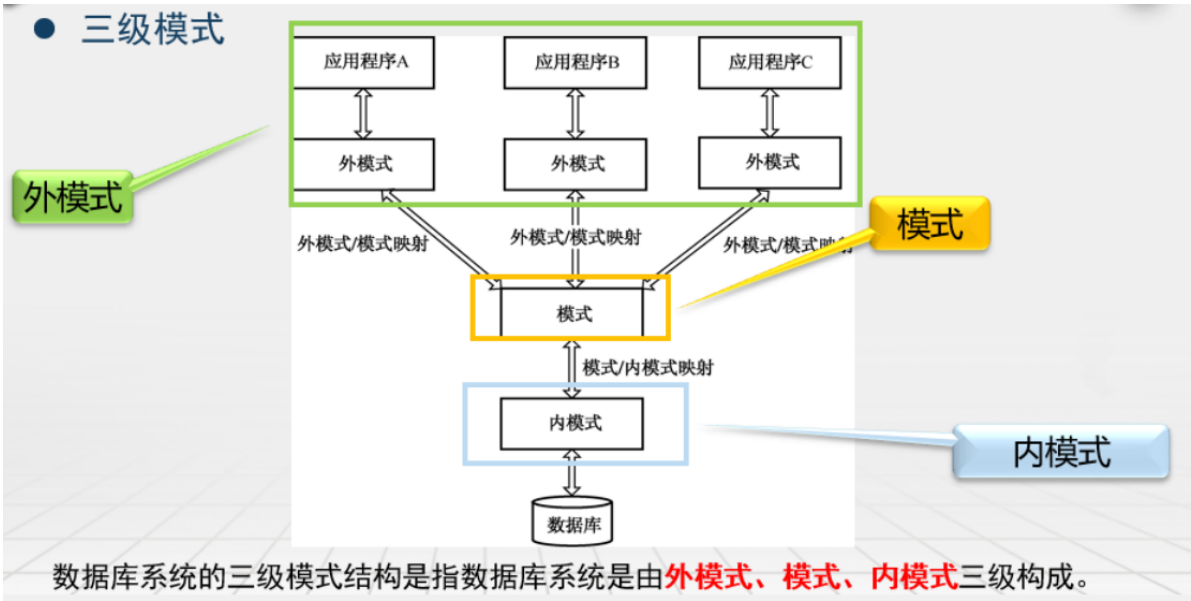
数据库系统是在文件系统的基础上发展起来的

数据模式（schema）

对数据库全体数据的逻辑结构和特征的描述

- 实例（Instance）：反映某一时刻数据库的具体状态，是整体的数据（不是一条），随着数据库的更新而变动。（实体，对应数据库里的一条记录，关系的一行）
- 数据库的三级模式结构：Conceptual schema（数据库的总框架，与物理细节和软件实现无关，DBMS提供数据定义语言DDL来描述逻辑模式），external schema（是模式的子集，是展示给用户的数据视图，保障安全性，也称子模式或用户模式），Internal Schema（描述数据存储的全部细节，包括是否加密、压缩、索引、存储路径）

  一个数据库只有一个模式，一个内模式，可以有多个外模式



- 二级映像与独立性

  逻辑数据独立性，当模式改变（不需要该用户知道的变化）时，DBA修改有关的外模式 / 模式映像，使外模式保持不变

  物理数据独立性，当数据库的内模式改变了（例如选用了另一种存储结构），数据库管理员修改模式 / 内模式映像，使模式保持不变

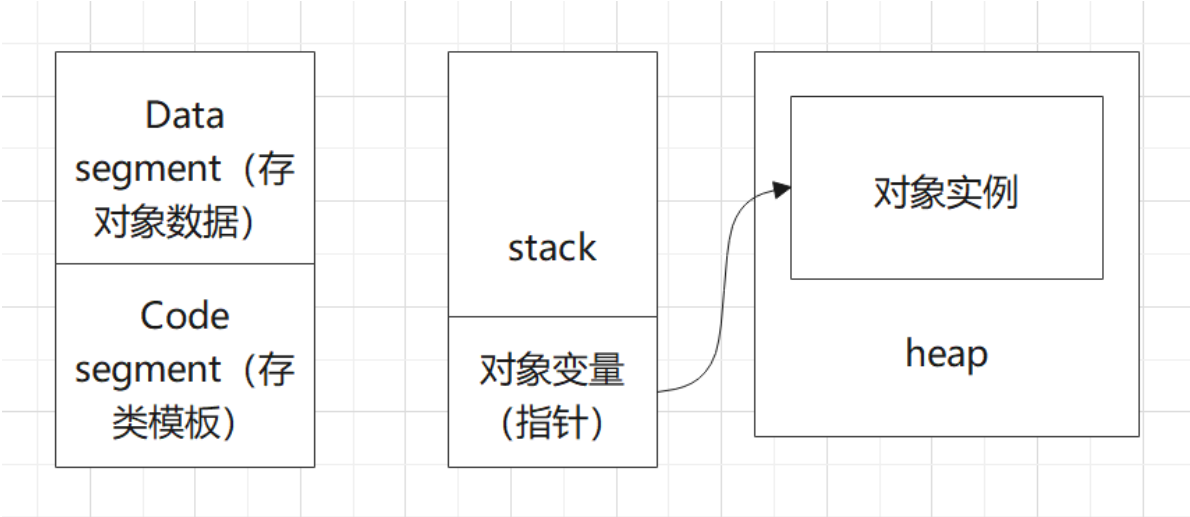
数据模型（model）

对现实世界数据特征的抽象，组成要素是数据结构、数据操作和数据的完整性（约束条件，一些数据只能在一定的范围内取值）

- 概念模型：面向现实世界建模，主要用来描述现实世界的概念化结构，与具体的DBMS无关，**概念模型必须转换成逻辑模型，才能在DBMS中实现**；常用实体（E）-关系（R）模型，与实体对应的数据库术语是记录
- 逻辑模型：面向用户建模，用户从数据库所看到的数据模型；

  层次模型（树状结构），网状模型（图），关系模型（二维表），面向对象模型，逻辑和物理一般是对应
- 物理模型：面向具体的DBMS，也面向机器，描述数据在存储介质上的组织结构

  对象在程序语言中的存储如下图：对象一般不直接在数据库中存储，而是用ORM，object relation mapping 把对象映射成关系来存储



建立数据库，要先抽象出数据模型，再构造出模式，从这个角度来讲，可以说模型包含模式，数据库结构的基础是数据模型

关系模型

关系是笛卡儿积（集合->元组）的有一定意义的、有限的子集

关系数据库的语言有：关系代数语言，关系演算语言，具有关系代数和关系演算双重特点的SQL语言

数据结构

关系实例是一个二维表，表中的每一列称为属性（attribute，field），每一行称为一个元组（tuple，record），表示一个实体，属性的取值范围称为域（domain），每一个格子叫做一个分量，属性的个数（列数）称为关系的目或度，能唯一区分不同元组的属性或属性组合称为键（key），候选键（唯一性、最小性：不是属性的数目最小，而是少了任意一个属性都不能构成码，所以可能存在一个三属性的组合和一个单属性同时是候选码的情况，不满足最小性的键被称为超键），主键（从候选码中选出来一个），外键：关系R中某个属性或属性组合并非该关系的键F，但F却是另一个关系S的主键，则称F为关系R的外键，关系R称为参照关系，S称为被参照关系，R和S可以是同一个或者不同的关系，外键可以体现关系之间的联系。

关系模式：

- 关系名
- 关系中属性的名字及相关联的域名
- 关系模式的表示方法R(U, D, DOM, F)、R(U, F)、R(U)（F:函数依赖，D：属性的域，DOM：属性向域的映射关系）

规范条件：

- 第一范式：属性不可再分
- 属性不能重名，行、列的顺序可以改变

完整性约束

- 实体完整性：主键对应的主属性的值都不能为空值（如果某个属性对应的值有空值，该属性就不能被纳入候选码了，可以被纳入码（超码），因为都取空值就无法区分），每个关系也必须有主键，也就是每一行都要有唯一的标志
- 参照完整性：如果指定了外键，则外键取值必须为：1）空值，2）被参照关系表中某个元组的主键值，体现了表间主键和外键之间的引用规则
- 自定义完整性：用户定义，数据库维护

关系操作

关系代数是以**集合运算**为基础的运算

关系数据库一定能实现关系运算

所有的查询操作都不会改变原本的关系，相当于是copy出了一个副本，在副本上操作

传统的集合运算：

- 并、交、差操作是水平方向的，不改变列，两个关系的属性的个数必须相同，属性名可以不同，只要对应属性的域相同就可以了

- n列a行和m列b行的关系R和S的笛卡尔积是一个（n+m）列，ab行的元组的集合，如果两个关系中有同名的列，把它们当做不同的来做，为避免列重名可以重新命名

PS: 交集可以通过差集表示，并、差、笛卡尔积是必不可少的，基本的关系运算：并、差、笛卡尔、选择、投影，其他都可以用基本运算表示

$$R \cap S \equiv R - (R - S) \equiv S - (S - R)$$

专门的关系运算：

- 从关系中找出满足给定条件的所有元组（行）称为选择，σ，用列名或者列数来筛选，可以用比较运算和逻辑运算（与∧，或∨），选择的结果与原关系模式相同，列不变
- 投影π，选出列，然后删除重复，因此行可能会变化
- 连接：在笛卡尔积上进行选择运算，连接运算并没有增加关系代数的表达能力，可以用其他关系运算符改写连接运算
    - 条件连接：在笛卡尔积上筛选出满足条件的属性
        - 等值连接，条件是属性（可任意规定，无需同名）值相等，不满足连接条件的tuple会被舍弃，没有空值
    - 自然连接，R和S**有同名的属性列**，做笛卡尔积，选择出同名列的值相等的部分，然后删除重复的列，**如果两个关系没有公共属性，自然连接就是笛卡尔积**，有多个同名的列就必须全部等值才能连
        - 半连接，自然连接在第一个操作数上的投影
    - 内连接，先做笛卡尔积，删除不满足条件的列；外连接，把相同属性值不相等的部分换成空值，然后删除重复的元组
        - R左外连接(LJN)S：所有来自R的元组和那些连接字段相等处的S的元组。
        - R右外连接S：所有来自S的元组和那些连接字段相等处的R的元组。
        - R全外连接S：左外连接和右边连接的并集。
- 重命名：$\rho_{表名(列名)}关系$，为了一些运算中不发生属性名和表名的混乱，比如合并医生和患者的名字，就把他们的名字属性都命名为name（只有一列可以不写表名），$\rho_{name}(\pi_{Dname}Doctor) \cup \rho_{name}(\pi_{Pname}Patient))$；自连接的时候，我们要找出同一部门的医生姓名对，重命名一张医生信息表，做笛卡尔积后筛选出同一个部门，而且注意no1和no2不能重名，一先一后的也要删除，所以可以选择no1>no2
- 除法：象集

  R÷S，R(X,Y), S(X,Y)，属性的交集是Y，找Y在X上的象集(写出x_1对应的所有y的集合)，写出象集包含了Y的所有属性值的X的属性值集合

  ◆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。

  $$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_Y(S) \subseteq Y_x\}$$

  $R \div S = \pi_X R - \pi_X(\pi_X R \times \pi_Y S - R)$，$\pi_X(\pi_X R \times \pi_Y S - R)$是没有所有Y的X

运算效率：

除、投影、选择、笛卡尔积中花费时间最长的是笛卡尔积

所以可以尽早做选择运算，在不损失的情况下早做投影运算，较少笛卡尔积的运算量，如果选最快的，其他都一样的前提下，选避开笛卡尔积的

等价变换规则：

并交差运算符合集合的运算律

两次选择运算的顺序可以交换，两次选择等价于条件用且连接的一次选择

笛卡尔积、条件连接、自然连接满足交换律R∞S=S∞R，满足结合律(R∞S)∞P=R∞(S∞P)，自然连接时可能出现没有相同属性的情况，那就是笛卡尔积，不影响后续的运算

如果F中涉及的属性都是E_1中的属性，则

$$\sigma_F(E_1\times E_2)=\sigma_F(E_1)\times E_2$$

如果$F=F_1\wedge F_2$，并且F_1只涉及E_1中的属性，F_2只涉及E_2中的属性，则由上面的等价变换规则1，4，6可推出：

$$\sigma_F(E_1\times E_2)\equiv\sigma_{F_1}(E_1)\times\sigma_{F_2}(E_2)$$

若F_1只涉及E_1中的属性，F_2涉及E_1和E_2两者的属性，则仍有

$$\sigma_F(E_1\times E_2)\equiv\sigma_{F_2}(\sigma_{F_1}(E_1)\times E_2)$$

它使部分选择在笛卡尔积前先做。

T（写关系代数表达式）基础步骤：

1. 审题，标出需要的属性（所在的表），关键字，结果要的属性
2. 把包含需要的列的关系做连接，自连接，做笛卡尔积或者等值连接，一定要先重命名表和列，确保都不重复，自然连接可以让想连起来的列重名，其他重命名
3. 筛选
4. 投影出用户要看的几列

关键字：**至少一次**，可以考虑做投影去重；**没有**...，减法；**至少n**，除开需要相等的列，把其他列重命名，然后自连接，然后用不等条件选择，最后投影去重；**恰好n**，至少(n+1)-至少n

！！注意没有不能用不等于条件来表示"没有"，一个厂商可以使用多个零件，每个零件都有生产地，找出没有使用天津生产的零件的生产商，city<>'天津'，是使用的零件不全是天津生产的

除非特别说明，ID才是唯一的标识，名字是有可能重复的

所有A的B

1. 找出A对应的属性作为Y，B对应的属性作为X
2. X为主键的关系作为除数，X作为外键的关系在X和Y上的投影作为被除数
3. $\pi_{X,Y}R\div S$

被除数没必要做投影，除法里面包含了投影，被除数必须先投影到X,Y上再除法，不能换顺序（如果X对应的其他属性不唯一，先除法再投影就会出错）

扩展的关系运算：

- 广义投影：投影的时候可以加入函数
- 聚集和分组（不是基础的，不要在关系代数表达式题目里面用）

  聚集函数（aggregate function）：输入值的一个汇集，将单一值作为结果返回。E.g. 聚集函数sum、avg、count、min、max。

$$\mathcal{G}\ F1(A1),\ F2(A2),\ \ldots,\ Fm(Am)(E)$$

也可以对一组元组集合（而不是单个元组集合）执行聚集函数，结果包含分组属性值和聚集函数的结果。也就是先按G1,G2,...,Gn分组，再求聚集函数的值

$$G1,\ G2,\ \ldots,\ Gn\mathcal{G}\ F1(A1),\ F2(A2),\ \ldots,\ Fm(Am)(E)$$

不能被代替的关系代数运算：∪， - ，×，σ，π，ρ（连接、除法、交集可以被代替）

- 递归闭包

数据库的设计

设计的阶段：需求分析->概念设计->逻辑设计->物理设计->实现->运行和维护（新奥尔良只包含前面四步）

结构设计（静态模型设计，包括概念、逻辑、物理设计）和行为设计（动态）

逻辑设计转物理设计可以直接用软件生成

设计方法：自顶向下，自底向上（通过分析用户的子需求，首先构建起局部概念模式，然后再向上组合成全局模式），逐步扩张，混合策略

需求分析：在数据库设计中，表示用户业务流程的常用方法是设计数据流图(DFD)、数据字典（DD）

数据字典可以用SQL语言查询

需求分析和概念设计之间的桥梁是DFD，概念设计和逻辑设计之间的桥梁是E-R图

调查的内容：信息需求、处理需求、安全性和完整性需求

数据抽象：分类，聚集，概括（继承性，比如学生父类，研究生和本科生子类）

数据库设计属于软件工程范畴

概念设计（E-R模型）

概念结构主要反映组织机构的信息需求

实体--方框（这里的实体指的是实体集合）

- 强实体：每个实例都能被实体集的主键唯一标识
- 弱实体：不能用实体集的属性唯一标识，不能独立存在，必须依赖于强实体集，比如电话号码必须依存于人

  一个弱实体集总可以通过往自己的属性中加入其标识实体集的主码属性而变成一个强实体集，但是弱实体集的主键本来就可以从它与强实体集的关系得到，这样添加后，它们将同时存在于实体集和关系集，产生冗余，所以一般不这样做

属性（和实体相连）--椭圆形框，实体的主键--下划线

- 名词能够作为属性的优先作为属性，满足两条准则就可以作为属性：

1. 不能再有其他的名词来描述它，不可再分
2. 一个属性不能对应两个实体（一个圆不能连两个方框）

- 简单属性和复合属性，在复合属性的基础上再分（树状）成简单属性，然后只保留简单属性
- 多值属性（双框），如果取值的个数确定，就再分后只保留子属性，值不好确定，就把属性升级成弱实体（弱实体和它与实体之间的联系都用双框）
- 派生属性（虚线框），之后尽可能消除，因为很容易数据不一致

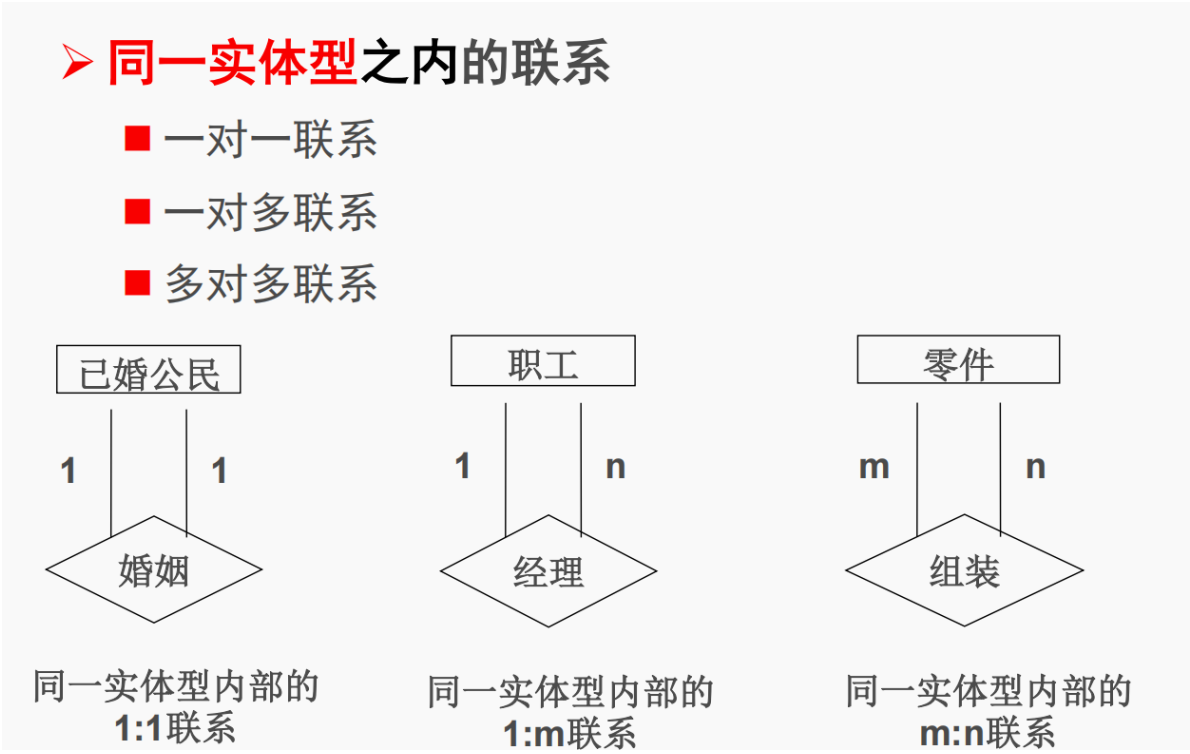
联系（连在属性之间）--菱形，联系也可能有属性

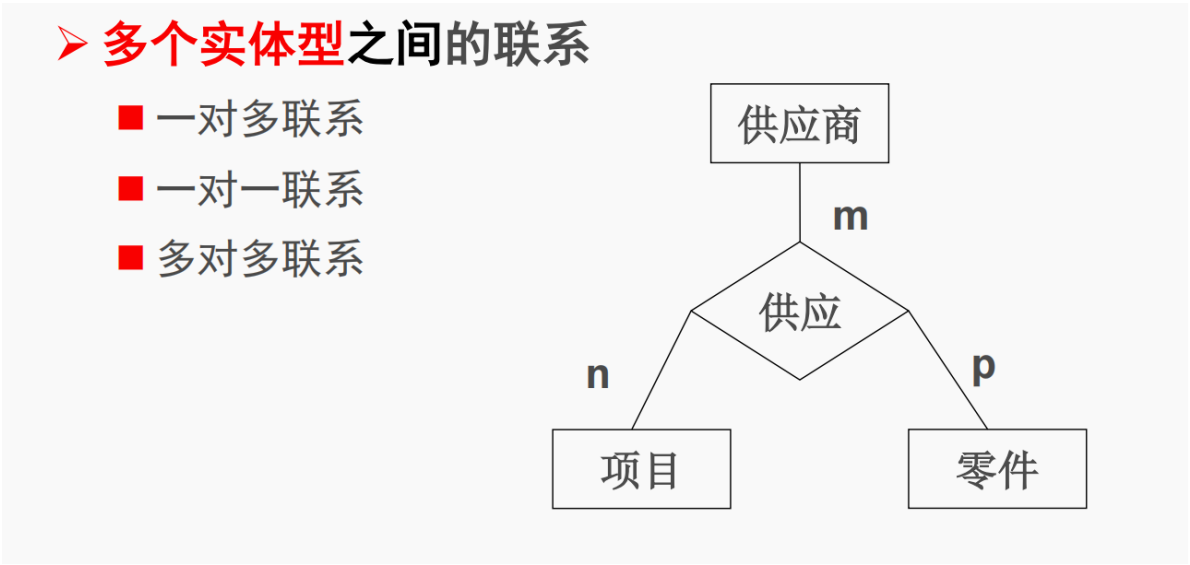
- 映射基数：一个实体通过一个联系可以与多少个实体相关联，四种

  联系可以是一对多的，也可以是一对一、多对多(m:n)的（E-R图中，1代表0到1，最多只有一个，多代表0到m，但是有一些关系中的实体个数必须是1不能是0，complete 1 to 1，否则关系就不存在了，比如婚姻关系一定是两个人的）

- 参与约束，如果实体集A中的每个实体都参与到与B的一个联系中，则称A全部参与到B中

联系一般是在两个实体之间，但是也有同一实体之内和多实体之间的联系，n元联系的一致性比较难维护





E-R模型的建立：先局部设计，然后合并，合并的时候要解决冲突，消除冗余，模式重构

冲突：

- 属性冲突：属性的域和取值单位冲突
- 命名冲突：不同图中名字和意思对应的不同，同名异义和异名同义
- 结构冲突：统一成实体或属性，同一实体包含的属性要取多个E-R图中属性的并集

可以添加不同局部E-R实体之间的联系，不能删除联系（只能修改，换一种方式来表示联系，不能删除）

逻辑设计（E-R转换）

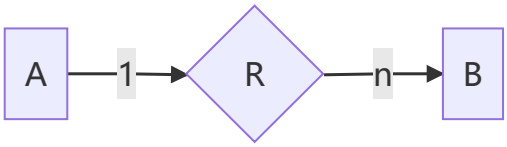
实体转换规则：将一个实体转换为一个关系模型，实体的属性就是关系的属性，而实体的键就是关系的键。

联系转换规则：实体之间的联系转换为关系模型，联系的属性直接转换为关系的属性。

与联系相连的实体的键转化为关系模型时，则分下述3种情况考虑

1:1的联系：可以转换为一个独立的关系模型，也可以与任意一端对应的关系模型合并

1:n的联系：可以转换为一个独立的关系模型，也可以**与n端**所对应的关系模型合并。



（实际的E-R图没有箭头）1个A可对应n个B，但1个B只能对应1个A，也就是B与关系R一一对应，可以合并，而A不可以。

m:n联系：转换为一个独立的关系模型。

注意任何一个能表示联系的关系必须将有联系的实体的主键纳入关系的属性，如果转化为一个独立的关系，联系本身的属性均转换为关系的属性，还要加入两个实体的主键，如果合并要加入另一端的主键，1:1时可以转化成独立或与任意一端合并，与之相连的每个实体的键都可作为新关系的候选键，1：n时可以独立或与n端合并，n端实体的键可作为主键，A和B组合不能作为主键，满足唯一性不满足最小，m:n时各个相连实体的键的组合不一定能作为关系的键，可能还要加入日期等标识，或者直接有编号属性来作为主键。

关系模式的逻辑设计要尽量消除数据冗余，冗余可能会带来操作（插入、删除、更新）异常（数据不一致）

标注主键的时候，主键是属性组合要连起来画横线，分开的话就表示有多个主键了

模式规范化

规范化是为了找出合适的数据逻辑结构

数据依赖

- 关系模式的五元组表示： R(U, D, DOM, F)
  - R：关系名
  - U：组成该关系的属性名集合
  - D：属性组U中属性所来自的域
  - DOM：属性向域的映象集合
  - F：属性间数据的依赖关系集合
- 关系模式的简化三元组表示： R（U, F）

函数依赖（FD）

X决定Y，Y依赖于X，X->Y：对给定的R(U), X含于U，Y含于U，任意t，s∈R，如果t[X]=s[X]，则有t[Y]=s[Y]（X确定后Y一定是确定的）

- **平凡依赖**，X包含Y，则一定有X->Y，自己依赖自己，不反映新的语义
  - 完全非平凡(completely)，X∩Y=∅（若X∩Y≠∅，X->X∩Y, X->Y-X，所以有X->Y）
- **完全依赖**(full)：如果X->Y，且不存在X'真含于X，使X'->Y，则称Y完全函数依赖(f)于X，否则称Y**部分依赖**(p)于X
- **传递依赖**，注意X和Y不能相互依赖，否则Z就是**直接依赖**于X和Y，不是传递依赖

逻辑蕴含：依赖函数集F能推导出依赖函数集G，即F逻辑蕴含G，F|=G

闭包（closure）：函数依赖集合F所逻辑蕴含的函数依赖全体称为F的闭包（完备集），F^+

Armstrong公理：

- 自反性：就是平凡函数依赖，一定依赖于包含自己的属性组合
- 增广性：X->Y, 则有X∪Z->Y∪Z （反过来不成立，可能X不能决定Y，但Z能决定，也可以实现XZ->YZ)
- 传递性

正确：从FD集（对给定的属性，任意的函数依赖组成的集合）中理规则推出的FD必定在F^+中

完备：F^+中的FD都能从*F*集使用推理规则推出

T：证明

1. 用函数依赖的定义
2. 用Armstrong公理，一般先增广再传递，常用合并性

属性集的闭包：属性集能决定的所有属性的集合

X^+={A|A∈U且X->A在F^+中}

X->Y能用FD规则推导出的充分必要条件是Y真含于X^+

T求闭包：求X^+先把X本身写入闭包中，再加入闭包集中的属性能决定的属性，直到不能再增加

已知F求F+是一个NPC问题，但已知X求X+可在多项式时间内完成

闭包和键

R(U,F)，A^+=U则A是R的键

K是R(U,F)的候选码（minimal key），则U完全依赖于K(与唯一和最小的性质相符)

包含在任何**候选码**中的属性称为主属性

全码：所有属性的组合才能作为码

候选码唯一地确定一个元组，也就是说候选码决定每个属性，由合并性可知，这两个说法是等价的

T求minimal key：（！仅针对题目给出的条件）

计算机求key的算法，从单个属性到属性组合，依次求闭包，不计算超码

简单方法：如果**没有多个候选码，也就是不存在函数依赖连成环的情况下**，没有出现在箭头右边的一定是在minimal key里面的，只出现在右边的不可能是主属性

要注意检查一下是否可能存在多个主码！

FD集的最小依赖集

四个条件：$F_{min}^{+}=F^{+}$（FD完整），F_{min}^{+}中少一个FD闭包就会变化（FD不冗余），右边都是单属性，左边没有冗余的属性（右边对左边完全依赖）

如果两个FD集的闭包是相等的（若F含于G^{+}，G含于F^{+}，则$G^{+}=F^{+}$），那么它们就是等价的，相互覆盖的

任一FD集都能被右边是单属性的FD集覆盖（分解性）

F_{min}^{+}一定存在可能不唯一

T 计算函数依赖集F的最小依赖集G：

分解,G中每个FD的右边均为单属性->在G的每个FD中消除左边冗余的属性->在G中消除冗余的FD。?后面两步要交替进行，直至不能再化简(右边->左边->FD，顺序不能改)

模式分解

设计包含所有信息的大模式，根据一定规则将大模式分解成若干小模式，分解后的小模式满足范式要求

数据等价：用"无损分解"衡量

依赖等价：两个数据库模式应具有相同的依赖集闭包

无损分解：分解后的关系做自然连接后能还原（连接后增加或减少都叫作有损分解）

设 $\rho=\{R_1(U_1),R_2(U_2)\}$ 是关系模式 $R(U)$ 的一个分解，则 ρ 是无损分解的充分必要条件是：

$$(U_1\cap U_2)\rightarrow(U_1-U_2)\text{或}(U_1\cap U_2)\rightarrow(U_2-U_1)。$$

注意这里是属性集合的运算，不是关系的运算，交集就是都有的属性

差集可能有多个，满足其中一个即可

保持依赖

设 $\rho=\{R_1,\cdots,R_k\}$ 是 R 的一个分解，F 是 R 上的 FD 集，如果有 $\bigcup_{i=1}^{k}\prod R_i(F)\vdash F$，那么称分解 ρ 保持函数依赖集 F。

在每个关系中找依赖，注意原函数集的闭包集中所有**包含这几个属性的依赖**都要找到，有的可能是需要**推导**出来的，然后求并集，一条条的看原FD集中的依赖是否蕴含在并集中

范式

1NF：属性不可再分，1NF包含2NF

2NF：每个非主属性完全依赖于（每个）候选码

包含在任何候选码中的属性称为主属性，否则就是非主属性

候选码是单码或全码都一定符合2NF

注意主码是AB，ABC→E，这不是部分依赖，部分依赖一定是子集，超集是冗余的，最小依赖集会把它化简，AB->C|=AB->ABC，但是ABC->AB，这是直接依赖

T 2NF模式分解：

2NF分解算法：将关系模式R分解成2NF模式子集

- 设有关系模式R(U)，主键是W，R上还存在函数依赖X→Z，其中Z是非主属性和X ⊂W，则W→Z就是一个部分依赖。此时应该把R分解成两个模式：
- ① R_1（XZ），主键是X；
- ② R_2（U-Z），主键仍为W，外键是X（参考R_1）。
- 利用外键和主键的连接可以从R_1和R_2重新得到R。
- 如果R_1和R_2还不是2NF，则重复上述过程，一直到数据库模式中每一个关系模式都是2NF为止。

不满足2NF是因为候选码中的多个属性各决定了一部分其他的属性，所以可以拆开

3NF：每个非主属性都不传递依赖于（每个）候选码

2NF包含3NF，部分依赖可以看作是一种特殊的传递依赖，没有传递依赖就一定没有部分依赖，没有部分依赖不一定没有传递依赖（单码）

T 3NF模式分解：分解直接按3NF分解，不要分解成2NF，再3NF

1.计算R的候选码（最小）

2.找依赖X->Z, Z不含于X，X不是候选码（排除直接依赖和平凡依赖）

3.分解成R1(XZ)，R2(U-Z)

4.对分解出的两个关系重复以上三步，一定要重新检查，除非分解到只有两个属性了

注意如果有多个候选码，每次分解只针对一个，但是完成后要针对每个检查是否存在传递和部分依赖，存在就还要继续分

！！A->B,B->C，那么一定要先从最尾端的FD开始分解，如果按A->C分解就会损失中间的函数依赖

BCNF：每个属性都不传递依赖于候选码，3NF包含BCNF

◆关系模式*R*（Bno，Bname，Author）的属性分别表示书号、书名和作者名。假如每个书号只有一个书名，但不同的书号可以有相同的书名；每本书可以有多个作者合写，但要求每个作者参与编著的书名应该互不相同。

◆*R*上的FD如下：Bno→Bname和（Bname，Author）→Bno

◆因此*R*的关键码是（Bno，Author）或（Bname，Author），因而模式*R*的属性都是主属性，*R*是3NF模式。

◆但根据两个FD可知，属性Bname传递依赖于关键码（Bname，Author），因此*R*不是BCNF。

◆例如，一本书由多个作者编写时，其书名与书号之间的联系在关系中将多次出现，会导致数据冗余和操作异常。

T BCNF模式分解：

■ BCNF分解算法1：将*R*无损分解且保持依赖地分解成3NF模式集。

◆① 对于关系模式*R*和*R*上成立的FD集*F*，先求出*F*的最小依赖集，然后再把最小依赖集中那些左部相同的FD用合并性合并起来。

◆② 对最小依赖集中每个FD *X*→*Y*去构成一个模式(*XY*)。

◆③ 在构成的模式集中，如果每个模式都不包含*R*的候选码，那么把候选码作为一个模式放入模式集中。

这个算法并不保证能分解成BCNF，如果不能无损且保持依赖，那用这个算法之后就是保持不变

■ BCNF分解算法2：将*R*无损分解且保持依赖地分解成3NF模式集。

① 对于关系模式*R*和*R*上成立的FD集*F*，先求出*F*的最小依赖集。

② 对每一个不满足BCNF的函数依赖*X*→Z，将R分解为

R1(*X*，*Z*)，*X是主码*

R2（U-X），X是外码，主码需要计算得出。

③ 对于R1和R2，重复上述步骤直到所有模式满足BCNF。

例：设关系模式*R*（*ABCDE*），*R*的最小依赖集为{*A*→*B*，*C*→*D*}。从依赖集可知*R*的候选码为*ACE*。

算法2能保证分解后的模式是无损的，但不一定保证保持依赖

要求保持依赖，一定能达到3NF，但不一定能达到BCNF，要求无损，一定能达到3NF和BCNF，要求保持依赖又无损分解，也一定能达到3NF

BCNF中还是会有一些冗余，不能消除所有的操作异常

！！主/非主属性、范式、分解都是针对候选码来说的，范式必须对每个候选码都成立

推论：

R符合BCNF的充要条件：最小依赖集的每个函数依赖X->Y左边X都是**候选码**（都是候选码就不存在传递依赖，如果不是候选码一定存在部分或传递依赖，可以存在候选码依赖于候选码的情况，但候选码之间一定是相互依赖，不构成传递依赖）

R符合3NF的充要条件：最小依赖集的左部是候选码，或者右部是主属性

（如果不说最小依赖集，左部都是超码也可以保证BCNF）

如果只有一个候选码，满足3NF就一定满足BCNF

R(A, B)一定符合BCNF

全码一定符合BCNF（全码一定是唯一的候选码），单码一定符合2NF，不一定符合3NF，可能存在传递依赖，但是全码一定不存在传递依赖，如果存在就不会是全码了

！！每个属性都是主属性不等于全码，可能是有多个候选码，覆盖了所有属性，例如之前作家和书的例子，一定符合3NF，但未必符合BCNF

T判断是否R(U,F)属于XXNF：求主键，然后看有没有部分和传递依赖，更快的是用充要条件+几个特殊情况

综合T建立关系数据库：E-R图->关系模式->求最小依赖集->主码、外码

根据语义得到一些依赖关系，注意X->Y，X与Y是多对一，所以每个多对一关系都要转化成一个FD，还有一些组合关系的FD

如果没有画E-R图，转化成关系模型的时候注意不要漏掉了联系，1:1，1：n都可以与实体模型合并，多对多的关系可能需要另外加入

求最小函数依赖集，先一个个的找每个属性能决定的，然后再三步化简

半期考试提醒

两张表做自连接，检查表名、列名是否重名（笛卡尔积或者条件连接的时候列不能重名，自然连接可以，而且还可以改成相同的名字来使它们连接），重名就一定要重命名，表名不能重名

可能有重名的情况，一般能使用编号尽量使用编号当条件

字符串属性用' '，关系()

E-R设计题，注意根据语义识别，不要过度推断，注意标上联系的对应数量，联系转变成独立的关系或是合并都要**加入两端实体的主键**

注意传递依赖不能是平凡依赖或者第一个依赖是双向的