第一章　绪论　　**21071003 高立扬**

1. 数据

能够输入计算机并能被计算机程序识别和处理的符号集合

2. 数据库管理技术发展四阶段

人工管理阶段(1950)、文件系统阶段(1950-1969)、数据库系统阶段(1970)、多远数据库技术阶段(1990-)

3. 数据库(Database/DB)

是存储在计算机内的有组织、可共享的数据集合，可以看成是电子文件柜

4. **数据库中的数据满足**

①数据按一定的**数据模型**组织，在描述和存储上具有较小的冗余度　　②数据之间相互**联系**

③由**数据库管理系统**(DBMS)进行管理

5. 数据模型及三要素

- 数据的组织方式，不但表达数据本身，而且表达数据之间的联系，以及数据的一致性约束

①数据结构：组织方式　　②数据操作：对数据进行的各种操作及规则

③数据约束：数据本身及数据之间存在的依存及制约关系

- 历史上出现过的数据模型：**层次、网状、关系**、面向对象、对象关系、XML

6. 数据库管理系统

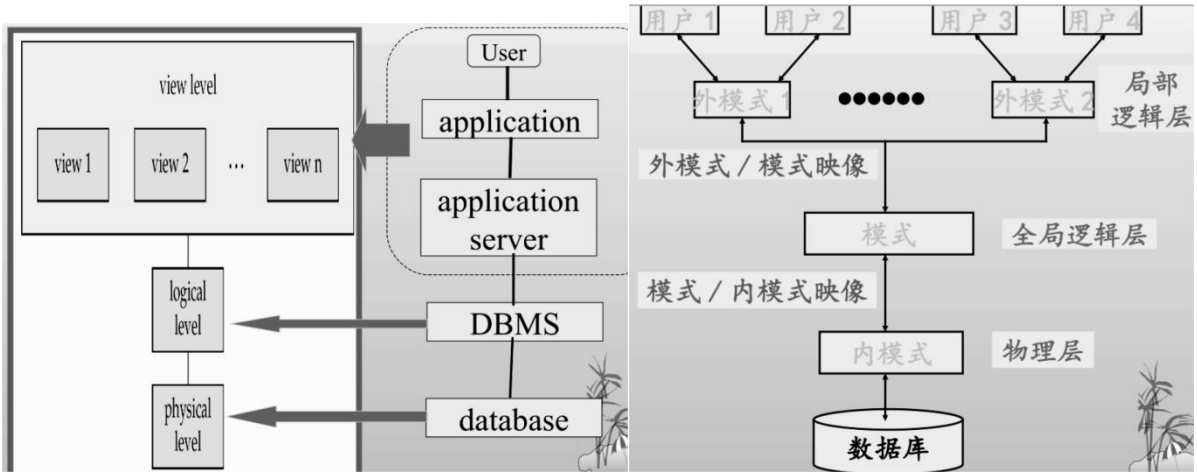
- 定义：对数据进行统一管理和控制的软件系统（介于用户和 OS 之间的系统软件，基于某数据模型）
- 目标：科学地组织和存储数据、高效地获取和维护数据。
- 功能：数据库的①定义功能②操纵功能③运行控制功能

7. 数据库系统 DBS

是 DB、DBMS、应用程序、DBA(数据库管理员)、计算机系统平台的总称

8. DBS 的三级结构

- 视图层：描述用户使用的数据结构
- 逻辑层：描述数据库的逻辑结构
- 物理层：①数据存储②存储路径③所占空间④填充率⑤存贮块大小⑥记录存储⑦索引⑧结构⑨存取



- 数据独立性：数据与应用程序在一定程度上相互分离
- 逻辑独立性：模式**改变**->外模式/模式**映像改变**->外模式**不变**->应用程序不变
- 物理独立性：内模式**改变**->模式/内模式**映像改变**->模式(尽量)**不变->外模式和应用程序不变**

9. DBA

对数据库系统（应用程序和数据）进行集中控制的人，他的职责如下：

①**模式(schema)定义**　②存取方式定义　③模式的修改　④用户授权　⑤日常维护

10. 文件系统&数据库系统

- 文件系统的优点①长期保存②物理&逻辑结构有简单区别③数据不再属于某程序
- 文件系统的缺点①**没统一数据模型，数据访问困难**②数据孤立③完整性问题④**更新的原子性问题**⑤并发异常⑥安全性问题
- 相比之下**数据库系统的优势**：①数据共享②减少不必要的冗余③较高数据独立性④加强数据安全性

第二章 关系模型

1. **关系模型**

用二维表（集合）来组织数据及数据之间联系的模型

（1）属性：每一列必须有名字，称为属性

（2）域：属性的取值集合

（3）元组：每一行数据。相当于一个记录值

（4）关系的特点

①关系表中每一个属性都是原子的

②各属性不能重名

③行列次序不重要，交换行列不影响语义

④不允许相同元组出现

（5）关系模式（instances）与关系实例（schemas）如右

①关系名与关系的属性集称为**关系模式（模式）**

②对应一个模式的当前元组集合，称为**关系实例（关系）**

③模式相对稳定，关系经常变动

（6）键码

定义：若关系中的某一属性（组），它的值能唯一的标识一个元组，且它任意一个**真子集不能标**识，那么就称他为关系的**键码/候选码/码/主码**

特点：**唯一性，最小性**

★相关重要概念★

①超码：以键码为**子集**的任意属性集

②**主属性：构成键码的属性**　　③非主属性：不构成键码的属性

④外码：关系 R 中的属性 X 是另一个关系 S 的键码，则 X 是 R 的外码（foreign key）。是表示关系之间联系的重要手段

（7）关系表示数据

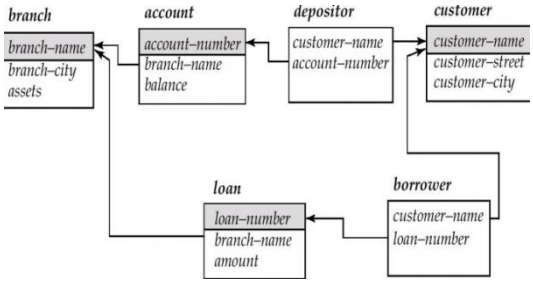
用关系模型组织数据时，按数据的特征分类，用一张二维表组织

（8）关系表示数据间联系

用关系模型表示两类数据的联系时，将能代表两类数据属性和描述联系性质的属性组成一张表

（9）关系数据库

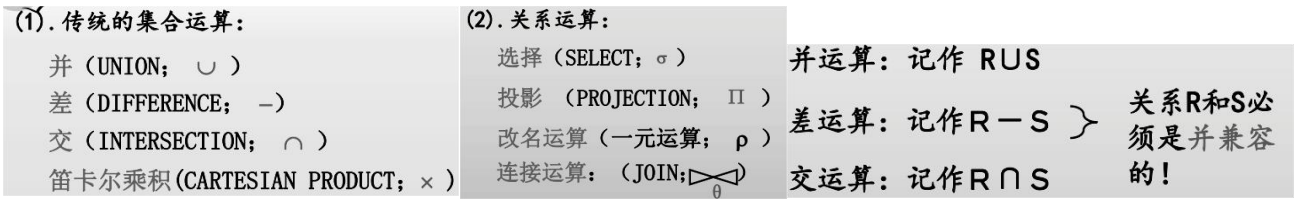
是若干关系构成的集合，一般来说，数据库中的数据组织方式用模式图来表示



2. 关系代数

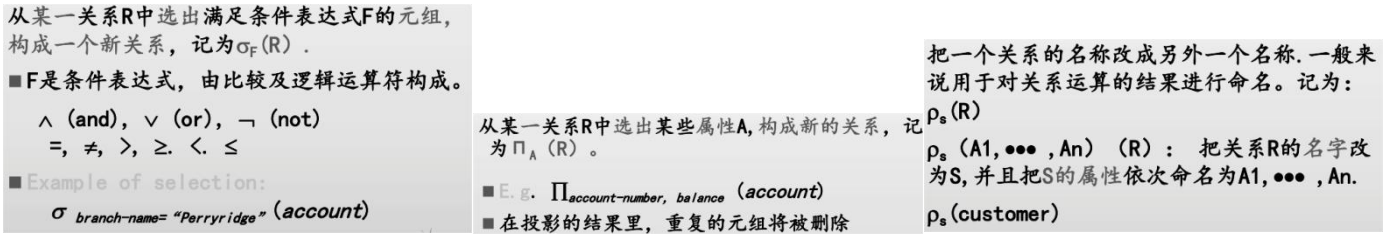
是**抽象的、过程化**的**查询语言**，特点是：运算**对象**和运算**结果**都是**关系**

（1）基本运算

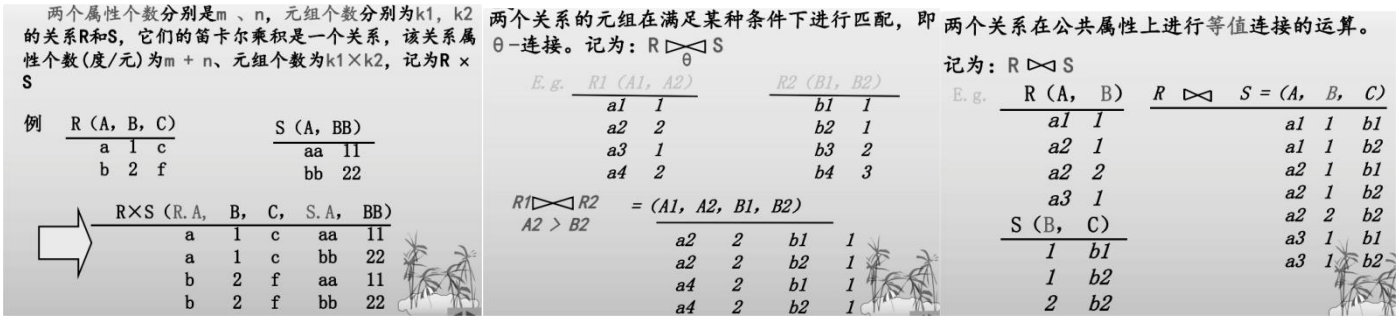


（2）并兼容问题：两个关系的属性个数相同，并且对应属性的域也相同，则称这两个关系并兼容。

（3）选择运算、投影、改名



（4）笛卡尔乘积、θ 连接、自然连接



（5）复合运算：注意灵活运用逻辑运算

eg.从 Movies(title,year,length,studio)关系里面选出由 Fox 公司制作的长度不少于 100 分钟的电影的名称和年份。Π title,year(σ studio=“Fox” ∧length>=100(Movies))

3. 关系的完整性

①实体完整性：保证关系中的每一个元组都是可识别且唯一的。

②参照完整性：一个表中的一个元组的存在以**另一个表中某一元组的存在为前提，外码实现**

③用户定义的完整性/域完整性/语义完整性：针对某一具体应用领域定义对数据的约束条件

作用：执行插入、删除、更新操作时检查完整性

第三章 数据库建模与实体联系模型

问题（目的）：如何将现实世界中的数据组织到数据库中

1. 数据库设计的基本步骤

（1）内容：构造较优的数据模式

（2）步骤：需求分析、设计概念模型、设计逻辑模型、设计物理结构、系统实现、试行&维护

（3）概念模型：从应用语义视角来抽取模型，按用户的观点来对数据和联系进行建模

特点是面向用户、面向现实世界，与 DBMS 无关

（4）表达工具：**实体-联系模型（E-R）**，语义对象模型

2. 实体-联系模型 E-R

由实体和联系构成的模型

（1）基本概念

①实体：现实世界中**客观存在**的一个具体或抽象的事物。能与其他实体**相互区分**。

- 属性：**用来描述实体特征的数据项**。
- 实体标识符：能够唯一标识一个实体的属性（组），且它的子集不能标识。

②实体集：具有共同属性的一类实体的集合。

- 属性类型：简单属性、组合属性、单值属性、多值属性、派生属性（由其他属性计算而来）

③联系：实体之间的相互关系

- 联系的属性：描述联系的特征
- 一个联系集的所有联系有相同的属性

④联系集：两个或多个实体集间，实体间的联系的总集合

- 在关系模型中，联系集可以用关系来表达：

{(e1, e2, … en) | e1 ∈ E1, e2 ∈ E2, …, en ∈ En}其中 Ei 表示实体集(e1, e2, …, en) 是一个元组

- Example: depositor 表示的是客户与帐户间的联系集，(Hayes, A-102) ∈ depositor
- 联系的度：一个联系集所涉及的实体集个数

涉及两个实体集的联系，叫**二元联系**

三个及以上，叫多元联系

⑤二元联系的类型：由一个实体集中的一个实体与另一个实体集中多少实体产生联系的角度，来定义

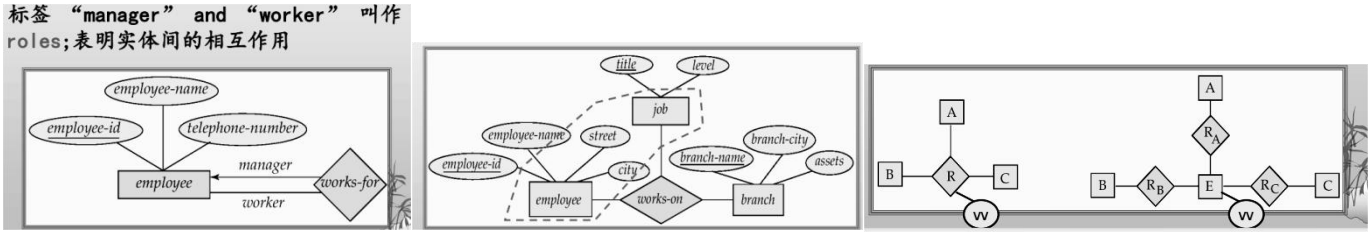
- 一对一，一对多，多对多

（2）E-R 图基本概念

①基本概念：

• 矩形表示实体、椭圆表示属性、菱形表示联系集、直线连接实体集与联系集、下划线说明键码
• “→”表示“一”，“—”表示“多”

②联系中的角色 roles：如果联系涉及的实体集是一个，需要区分它在联系中的不同作用



③多元联系联系类型判断：用（n-1）个实体集中的实体与另一个实体集中实体的联系类型来确定

④多元联系转化为二元联系：将 R 用实体集 E 来代替，然后生成以 E 与其它实体集间的二元联系: RA,.RB, RC 原联系 R 的属性成为 E 的属性

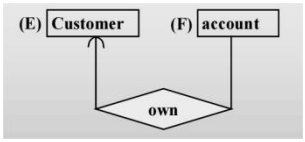
（3）E-R 图设计问题

①忠实性：以现实世界为唯一依据

②避免冗余：现实世界中的一个事物，在数据库中只能表达一次

③尽量简单：属性、实体集和联系从简，能不用就不用，能合并就合并

④选择合适的事物类型



（4）约束的建模

①主码的设置

②参照完整性约束：E 到 F 多对一联系，若 E 中存在一个实体 e，则 F 中必须存在一个对应的实体 f

③域约束：属性类型和取值范围的约束。在 E-R 图中没有特殊规定

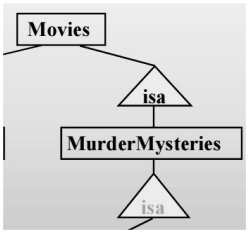
（5）子类实体

①对一个实体集进行细分或概括，就形成了高低层的实体集，分别称为**超类和子类**

②子类实体继承超类的所有属性和联系，同时也有属于自己的属性和联系

③在 E-R 图中用标有“ISA”的三角形与超类联系

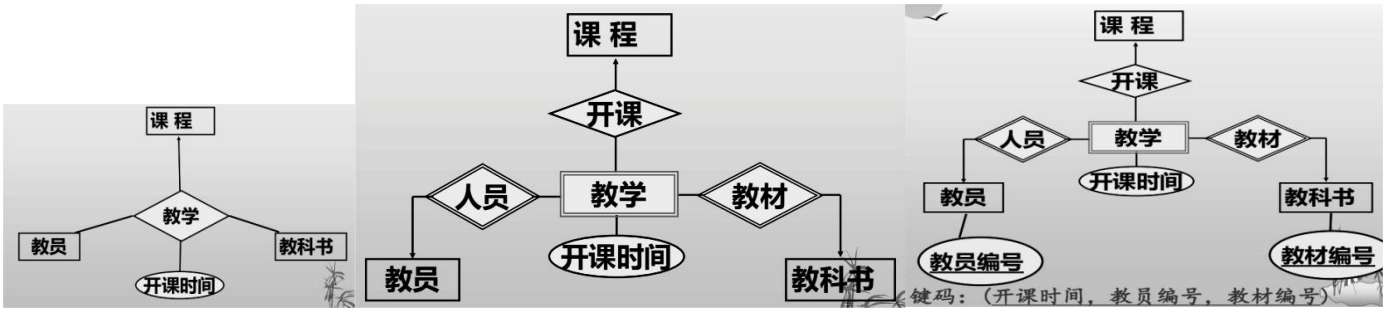
④当子类实体没有自己的属性和联系时，则其不出现在 E-R 图中!!!!



（6）弱实体集

①没有主码的实体集

②多远联系转化的连接实体集往往是弱实体集



③弱实体集的键码构成：自己的属性+多对一联系从其他强实体集获取主码

④弱实体集与为它提供键码的实体集间的联系用双线棱形

3. ★★★E-R 模型向关系模型的转化★★★

（1）实体集转化

①强实体集转化：实体集的主码和属性成为关系模式的主码和属性。

②弱实体集转化：自己的属性 + 为自己提供主码属性的强实体集的主码。

③子类实体集转化：

method1：父类主码 + 子类自己的属性

method2：子类继承的所有属性 + 自己拥有的属性

（2）联系集转化

• 原则：出现相同属性名应该改名 • 关系的键码由联系的类型决定

①多对多：联系自己的属性 + 参与联系的实体集的键码（这些实体集的键码**都作为新键码**）

②**一对多**

• 联系自己的属性 + “多”实体集的键码（**“多”的键码为新键码**）

• 可以不用单独转化，只需要在“多”的实体集属性中，加入联系的属性和“一”的键码

③弱实体集与它所依赖的强实体集间的联系没有必要转化（**“多”的键码为新键码**）

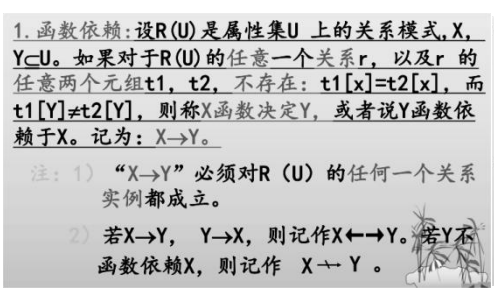
④一对一

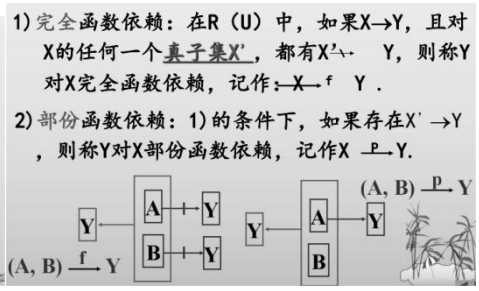
• 联系自己的属性 + 双“一”的键码（**任选一个作为新键码**）

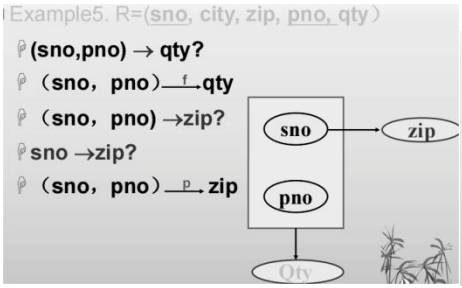
• 可以不用单独转化，合并两个“一”的所有属性，键码从原来各自的键码任选一个

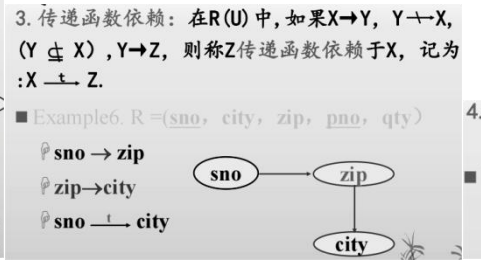
第四章 关系数据库设计理论

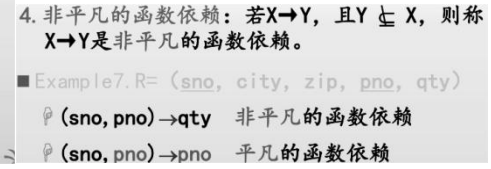
1. 函数依赖 ★★重点掌握★★ 函数依赖最高到 BCNF！！！









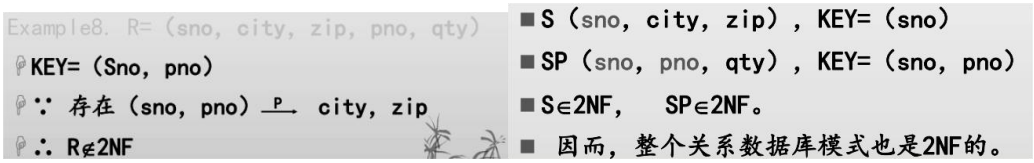


2. 范式

（1）定义：关系数据库中所有关系要满足的属性间依赖的要求

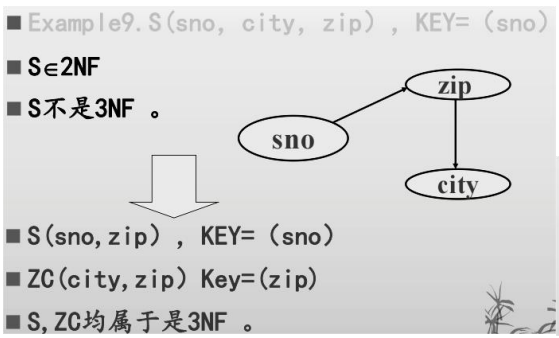
（2）1NF：各属性域必须是原子的

（3）2NF：R 是一个关系模式，若 R∈1NF，且每一个**非主属性完全函数依赖于主码**



（4）3NF：R∈2NF，且不存在键码 X、属性组 Y 以及非主属性 Z（Z 不包含于 Y）使得 X→Y（且反之不成立），Y→Z 成立，则 R∈3NF

人话：R 中**不存在非主属性对主码的传递函数依赖**



a. 所有非主属性完全依赖于每个键码。

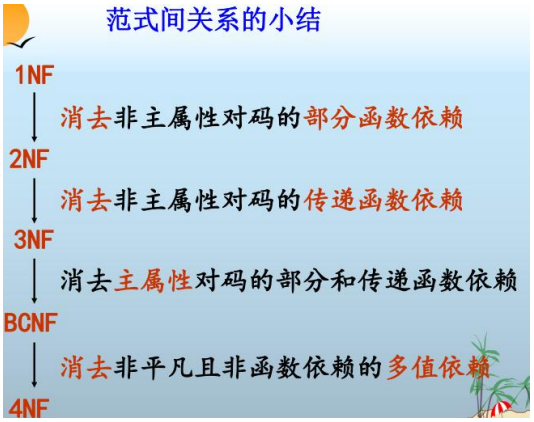
b. 所有主属性完全依赖于不包含它的键码。

c. 没有任何（主/非主）属性完全函数依赖于任一非主属性(组)。

（5）BCNF：R∈1NF，对于每一个 X→Y（Y 不属于 X），X 是超码。三个特点如上图↑

3. 多值依赖与 4NF

（1）多值依赖 MVD：R 是属性集 U 上的一个关系模式，X，Y 和 Z 都是 U 的子集,并且 Z = U-X-Y。多值依赖 X →→Y 成立，当且仅当对于关系模式 R 的任意一个关系 r，r 在 X 上的一个值对应唯一一组 Y 值，且这组 Y 值与 Z 的值无关。比如一门课可以对应唯一一组老师，和该课用哪本教材无关。



（2）结论

①对称性：X→→Y，则必有 X→→Z，当且仅当 Z=U-X-Y

②若 X→→Y，当 Y⊆X 或 Z=U-X-Y=Φ，则称为 X→→Y 为平凡的多值依赖。否则就是非平凡的 MVD

③传递性：若 X →→Y，Y→→Z（Z = U-X-Y）则 X →→Z。

（3）4NF：R∈1NF，如果对于 R 的每个非平凡 X→→Y，X 都是超码，则 R∈4NF。

（4）理解：非平凡多值依赖的一种约束!!!!，将每个非平凡的多值依赖左端限制为超码，实际上说明这个多值依赖是函数依赖！

4. 函数依赖集闭包

（1）函数依赖的逻辑蕴含：F 是 R 给定的一组函数依赖集合，若能用逻辑推理导出其他函数依赖，F 逻辑蕴涵这个导出的依赖　If F={A→B ，　B→C}, then we can infer that A→C　F⟹ A→C

（2）闭包：F 及由 F 所蕴含的所有的函数据依赖的集合，记作F^+

★★★★下面开始都是重点★★★★

5. 函数依赖 Armstrong 公理系统和推理规则

合并：若X→Y，X→Z，则 X→YZ

分解：若X→YZ，则 X→Y，X→Z

伪增广：若X→Y，W⊇Z，有XW→YZ

伪传递：由X→Y，WY→Z，有WX→Z.

Example15. $R = (A, B, C, G, H, I)$

$F = \{ A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

some members of *F*+

- $A \to H$
  - by transitivity from $A \to B$ and $B \to H$
- $AG \to I$
  - by augmenting $A \to C$ with G, to get $AG \to CG$ and then transitivity with $CG \to I$
- $CG \to HI$
  - from $CG \to H$ and $CG \to I$
- $AG \to H$
  - From $A \to C$ and $CG \to H$

6. 属性集 A 关于函数依赖集 F 的闭包运算

- F为属性集U上给定的一组函数依赖，a，β ⊆U。

则 a^+_F称为属性集a关于函数依赖集F的闭包。

- a^+_F = {β| a→β是所有由F给定和导出的函数依赖集合}。
- Example16. R（A，B，C），F={A→B，B→C}
  - ➢A^+_F ={ABC}

Example17. $R = (A, B, C, G, H, I)$

$F = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

$(AG)^+$

PROCEDURE:

1. ***result* = AG**
2. ***Result* = ABCG** $(A \to C$ and $A \to B)$
3. ***result* = ABCGH** $(CG \to H$ and $CG \subseteq AGBC)$
4. ***result* = ABCGHI** $(CG \to I$ and $CG \subseteq AGBCH)$

（1）确定关系模式的键码

- 如何确定属性组AG是键码(Is *AG* a candidate key)?
  - Does $AG \to R$? == Is $(AG)^+ \supseteq R$
  - Is any subset of AG a key?
    1. Does $A \to R$? == Is $(A)^+ \supseteq R$
    2. Does $G \to R$? == Is $(G)^+ \supseteq R$

Example19. $R = (C, G, S)$

$F = \{CS \to G\}$, Find all candidate keys for R

$(C)^+ = C$, $(G)^+ = G$, $(S)^+ = S$

$(CG)^+ = CG$, $(CS)^+ = CSG$, $(SG)^+ = SG$

$CS \to R$

STOP!

CS is a key of R!!!!

7. 键码求解，依赖集等价计算

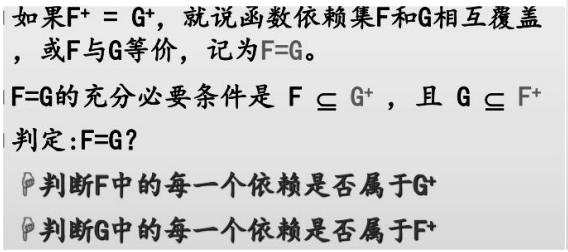
（1）键码求解

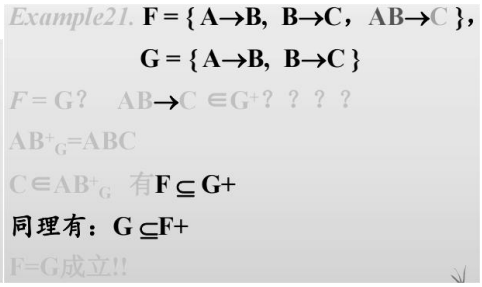
①只出现在函数依赖左边的属性，必定是构成键码的属性；
②只出现在函数依赖右边的属性，必定不是构成键码的属性；
③不出现在函数依赖中的属性，也必定是构成键码的属性；
④既出现在左边，也出现在右边的属性则可能是顾全构成键码的属性。

例1：给定关系模式R（X，Y，Z，U，V，W），依赖集F={X→Z，XY→Z，Z→UW，ZV→XY，ZU→W，VW→Z}，求：R的所有键码？

答案：R的键码有两个：XV和ZV

（2）依赖集等价





8. 模式分解

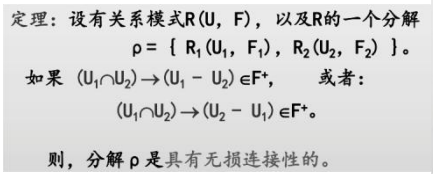
把一个关系模式 R 分解成若干个不相包含的关系模式{R1,R2…Rn}的操作

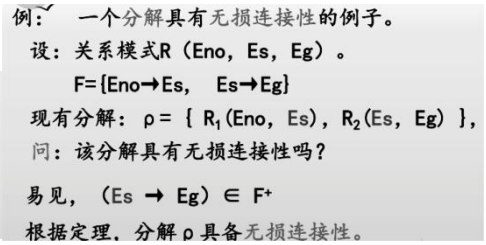
9. 关系投影&无损连接性判断

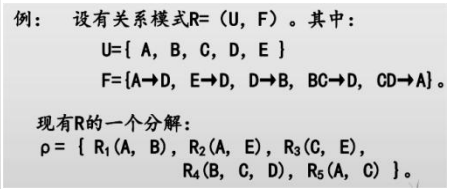
（1）投影举例 Decomposition of R = (Eno,Es,Eg) into R1 = (Eno,Es) and R2 = (Eno,Eg)

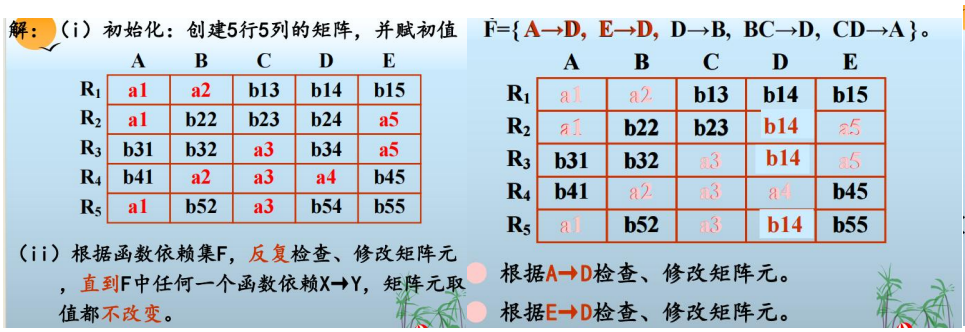
（2）无损连接 R=R1 U R2，$r = \prod R1(r) \prod R2(r)$

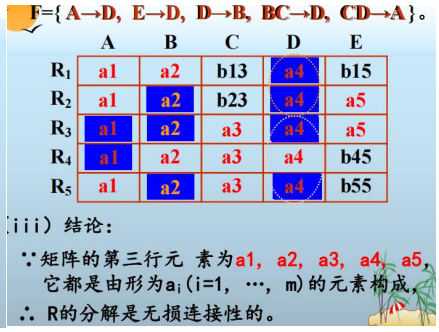
（3）判断——分解成两个或多个子模式











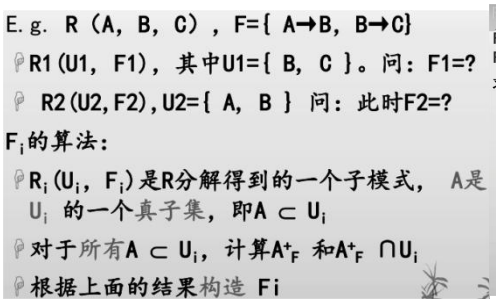
算法要点：

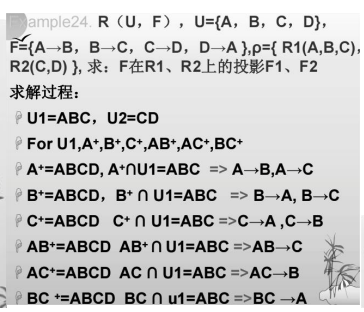
①反复使用函数依赖集，直到矩阵元都不改变。

②当某列上的一个 bij 改为 bkj(k＜j)或 aj 时，应该 把同一列、其它行上所有取值为 bij 的矩阵元 都要改成 bkj 或 aj 。

③使用函数依赖集检查、修改矩阵元之后，若矩 阵上存在一行，它都是由形为 ai(i =1，… ，m)的元素构成，则分解是无损连接性的；否则分解不具有无损连接性

10. 依赖的投影





F1 = {A→B, A→C, B→A, B→C, C→A, C→B, AB→C, AC→B, BC→A }

F1={A→B, B→C, C→A}

11. 保持函数依赖的分解

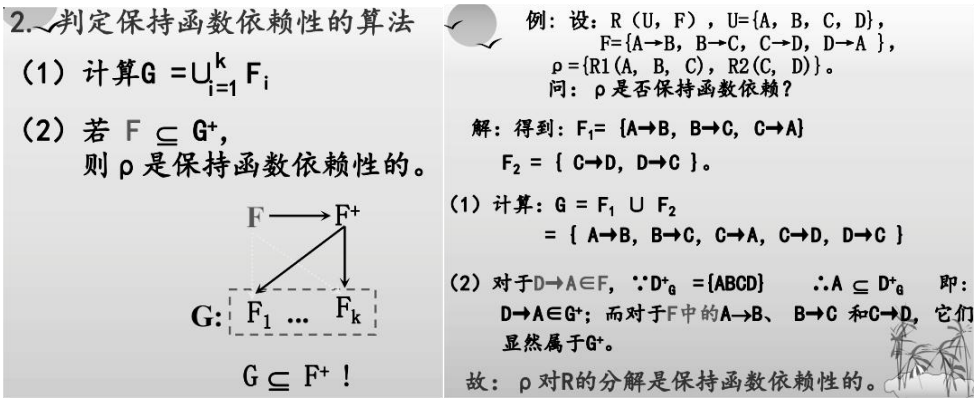
（1）概念 $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

（2）一些结论

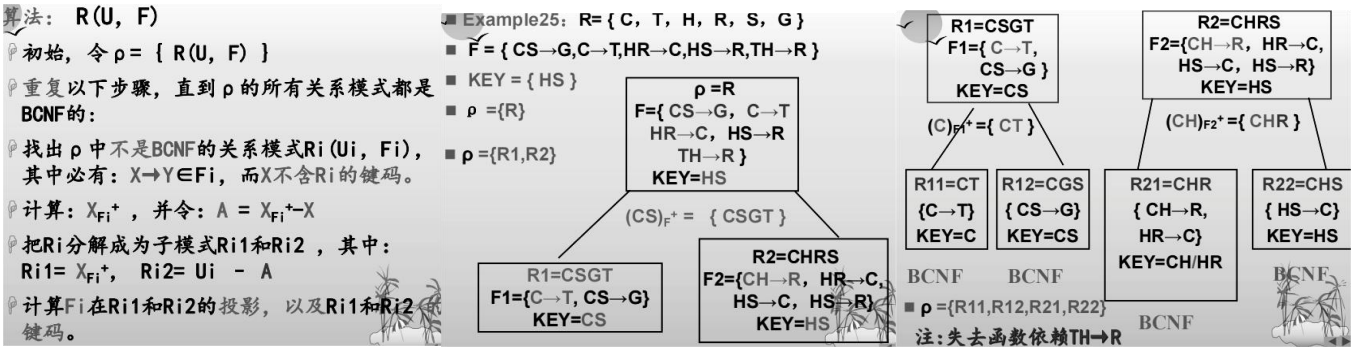
①R 一定可以无损连接性地分解成若干个 BCNF(甚至是 4NF）的子模式

②R 一定可以保持函数依赖性地分解成若干个 3NF 的子模式，但不一定能达到 BCNF。

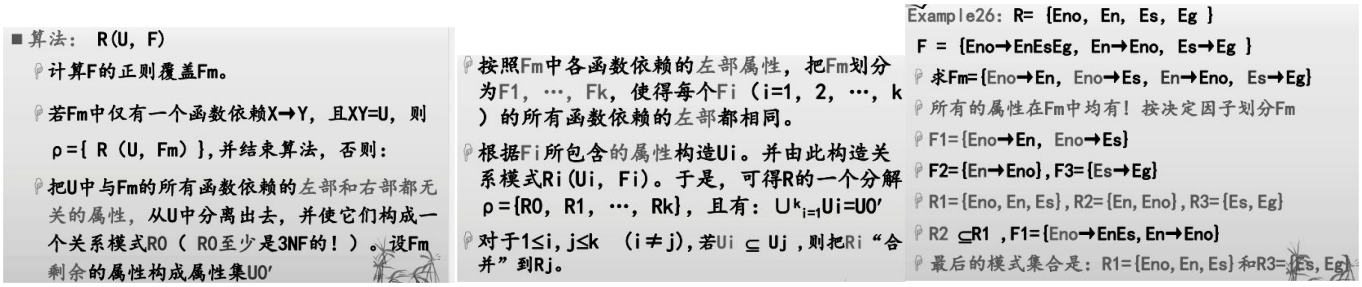
③R 可以既保持函数依赖性，又实现无损连接性地分解形成一些 3NF 的子模式，但不一定能分解而形成 BCNF 的子模式。



12. 保持依赖分解到 BCNF



13. 保持依赖分解到 3NF



第五章 SQL

1. SQL 特点

①综合统一②高度非过程化③面向集合的操作方法④一种语法结构，两种使用方式（自含式语言、嵌入式语言）⑤语言简洁、语法简单、易学易用

2. 基本表、查询表和视图表的特点和区别

（1）基本表：实际存在的表。即，基本表是实表。DBMS 保存并维护基本表的数据和元数据。在一个基本表上，可以创建并维护若干个索引表。

（2）查询表：用于存放查询的中间结果，以及查询结果的表。查询表是临时表。查询表的数据和元数据都是“临时”的。

（3）视图表：简称：视图。它是由基本表或其它视图表“导出”的表。视图表是“虚”表。视图本身没有独立存在的数据，没有实际的物理记录，逻辑存在，不占用物理空间。

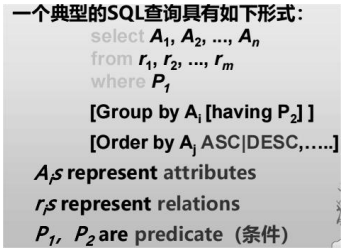
作用：从用户的角度隐藏数据，简化查询书写方式（**保护数据，简化查询**）

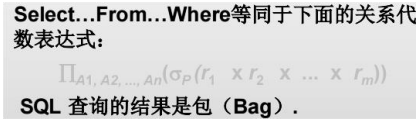
3. SQL 中的数据类型

char,varchar,int,smallint,numeric（存任意精度）,real（double precision）,float,NULL,date,time,timestamp

interval [时间间隔数量] [时间间隔单位]

4. 表创建、修改和删除 见 PPT 63-72
5. 索引的创建和删除 见 PPT 73、74
6. SQL 查询语句结构 如右
7. 空值概念





（1）概念：SQL 中用“null”表示信息缺失或不确定。用“ is null”进行空值测试

（2）运算：①与 NULL 相关的数学运算，结果都是 NULL ②NULL 与任意值比较返回 unknown
③聚集函数会忽略 NULL

（3）unknown：布尔值的一种，在 where 引出的条件子句中，如果条件返回的是 unknown，则认为是 false

- OR: (unknown or true) = true, (unknown or false) = unknown，(unknown or unknown) = unknown
- AND: (true and unknown) = unknown, (false and unknown) = false, (unknown and unknown) = unknown
- NOT: (not unknown) = unknown

8. 聚集函数（**sum, count, avg, max, min**） 见 PPT 30-36
9. 基本表的查询 **见 PPT 10-23** 字符串运算（**★选择题★**）**PPT 24、25** ORDER **PPT26** 集合运算（**并**）**PPT27-29**
10. **嵌套子查询 见 PPT40-56** **连接运算（忽略外连接）PPT 57-61**

第六章 SQL 约束

1. 键码（主码）约束 PK

PRIMARY KEY(属性（组）)，对应了实体的完整性约束

2. 单值约束 UNIQUE

在某条属性声明时写 UNIQUE，可以取 NULL 值

• PK 与 UNIQUE 的区别：

①一个表中 **PK 只能出现一次，UNIQUE 可以出现多次**：DBMS 在 PK 上建立聚族索引，而聚族索引的顺序与表中记录的存贮顺序一致。但在 Unique 上建立的是唯一索引,其与存贮顺序无关

②**UNIQUE 可以**取 **NULL 值，PK 不可以**

• PK 和 UNIQUE 的作用：

当对表进行**插入、修改**操作**时，DBMS** 将进行主码约束或单值**约束检查**

3. 取值约束 CHECK（全局&非全局）

（1）加在某属性末尾：对该属性取值约束

（2）全局约束（元组间约束）如右图

4. 参照完整性约束

若关系 R2 对 R1 有 REFERENCES，那么 R2 中的任意元组在外码 FK 上的取值，必须是 R1 上该码的取值，不能有 R1 该码没取过的值，或者取 NULL

- 保持参照完整性的**三个可选策略**

①任何破坏该性质的操作，都会被 DBMS **拒绝**

②**级联策略**：修改被参照的关系元组时，系统级联修改外码所在元组或外码属性值

③**置空策略**：修改被参照的关系元组时，系统修改外码属性值为 NULL

5. 断言

指数据库中数据要满足的条件。前面讲的各种约束都是断言的特殊形式。但断言的功能更强大，可以定义各种约束。create assertion <断言名>check(条件)

6. 触发器

触发器是一个语句集合（规则），当数据库作修改时，被系统自动执行。需要指定执行条件时机&采取动作。CREATE TRIGGER <NAME> BEFORE/AFTER <时机> [OF 属性] ON <表> [FOR EACH ROW] WHEN(...) [相关操作]

第七章 数据库的安全与控制

1. 数据库安全性概念

（1）概念：保护数据库，防止因用户**非法**使用数据库造成数据**泄露、更改或破坏**

（2）一般方法：①用户身份标识和鉴定②**存取权限控制**③定义并使用**视图**④审计制度⑤数据加密、

（3）存取权限基本类型：①读取②插入③更新④删除

（4）SQL 授权/收回

①授权：**GRANT <权限> ON <关系名或视图名> TO <用户> [WITH GRANT OPTION]**

注：<权限>例如 SELECT,INSERT； 最后面的[WITH...]仅限 owner 或 DBA 使用，由此获得“授权”权限的<用户>可以将获得的权限继续传递给其他用户。GRANT OPTION 也属于权限，可以被 owner 或 DBA 收回

②收回：**REVOKE <权限> ON <关系名或视图名> FROM <用户表> [CASCADE]**

注：CASCADE 是级联删除选项；如果要收回某用户授予 SELECT 的权限，需要在<权限>填写 GRANT OPTION FOR SELECT，注意 FOR 的使用

③局限性：SQL 不支持元组级别的权限控制，比如我们无法让 A 学生在成绩单中只看到 A 学生所在元组

2. 事务管理

（1）概念：事务是**访问**并可能**更新**数据库**数据**的一个**程序执行单位**

（2）性质 ACID

①原子性 Atomicity：组成事务的**操作不可分割**

②一致性 Consistency：一个事务必须是一个**正确程序**，使数据库从一个**一致状态**转为**另一个一致状态**

③隔离性 Isolation：一个正在执行的事务在**提交前，不允许对共享数据所做的改变提交给其他事务用**

④持久性 Durability：一旦事务**提交**，即使**系统发生故障，也不能丢失**该事务的**执行结果**

3. 事务的并发操作

（1）并发：多个事务**同时存取相同**的数据

（2）带来的问题：**数据一致性**问题①丢失修改②读入脏数据③不可重复的读

（3）封锁 ①排它锁 X：写锁 ②共享锁 S：读锁 仅 S 与 S 之间可以相容

（4）三级锁协议

①一级锁协议：事务在**修改数据之前**，必须先对其加 **X 锁**，直到事务**结束/提交/回退后**，才**释放**，**解决“丢失修改”**

②二级锁协议：在**一级锁的基础**上，事务在**读**取数据**之前**，必须先对其加 **S 锁**，**读入后立即释放**，**解决“读脏数据”**

③三级锁协议：在**一级锁的基础**上，事务在读数据之前，必先对其加 S 锁，直到事务**结束/提交/回退后**，才**释放**，**解决“不可重复读”**

（5）并发调度的可串行性

如果多个事务的**并发**调度执行过程的**结果**，**与**它们的一个**串行**执行**过程产生的结果相同**，则称并发调度是**可串行化**的。

（6）两阶段锁协议

每个事务分两个阶段提出锁操作①获得锁阶段：任何事务可以申请获得锁，但不能释放锁②释放锁阶段：任何事务可以申请释放锁，但不能获得新锁。如果**所有的事务都遵守“两段锁协议 ”，则这些事务是可串行化的。**