# 数据库复习提纲

题型

* **选择题： 15小题 30分**
* **判断题： 5小题 10分**
* **简答题： 4小题 20分**
* **分析题： 2小题 20分**
* **综合题： 5小题 20分**

第一章 DBS概述

* **四个概念：数据、数据库、数据库管理系统、数据库系统，相互之间的包含关系**

**1.数据(Data):**

**数据是信息的符号表示，或称载体**

**2.数据库(DB):**

**能长期存储的由DBMS统一管理的被多个用户所共享的数据集合**

**3.数据库管理系统(DBMS):**

**存取数据库中数据的一套程序,** 是**系统软件.**

**4.数据库系统(DBS):**

**带有数据库的整个计算机系统,包括硬件、软件、数据、人员**

* **三级模式：物理模式（内模式），逻辑模式（模式），和子模式（外模式）的含义**

**1.** **内模式(Storage Schema)：**

**又称存储模式，是数据的物理结构及存储方式。**

**2.** **模式(Schema)**

**所有用户的公共数据视图，是数据库中全体数据的全局逻辑结构和特性的描述**

**3.外模式(Sub-Schema)：**

**用户的数据视图，是数据的局部逻辑结构，模式的子集**

* **什么是两种数据独立性：逻辑数据独立性，物理数据独立性**

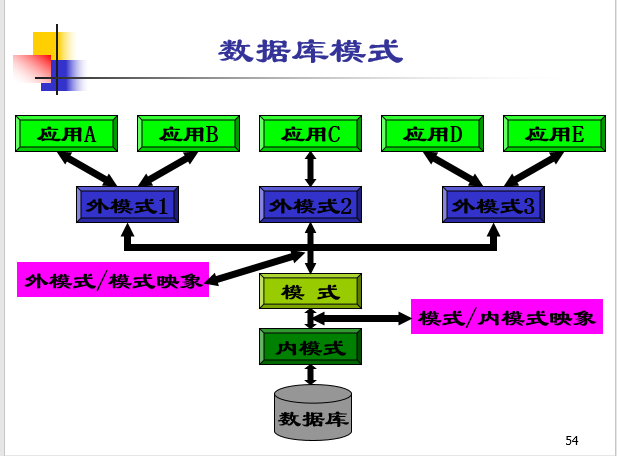
**1.逻辑数据独立性：**

**当数据的逻辑结构改变时，应用程序不必改变**

**2. 物理数据独立性：**

**当数据的存储结构改变时，应用程序不必改变**

* **三级模式两级映像以及如何实现两种数据独立性：逻辑数据独立性，物理数据独立性**



**1. 外模式/模式映象：**

**定义某一个外模式和模式之间的对应关系，映象定义通常包含在各外模式中，当模式改变时，修改外模式/模式映象，使外模式保持不变，从而应用程序可以保持不变，称为数据的逻辑独立性**

**2. 模式/内模式映象：**

**定义数据逻辑结构与存储结构之间的对应关系，存储结构改变时，修改模式/内模式映象，使模式保持不变，从而应用程序可以保持不变，称为数据的物理独立性**

* **数据库系统的组成。**

|  |  |
| --- | --- |
| C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687829592281.png | C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687829623072.png |

* **概念数据模型（定义、ER模型）**

**1.定义：**

**按用户的观点来对数据和信息建模，用于组织信息世界的概念，表现从现实世界中抽象出来的事物以及它们之间的联系。这类模型强调其语义表达能力，概念简单、清晰，易于用户理解，它是现实世界到信息世界的抽象，是用户与数据库设计人员之间进行交流的语言**

**2.E-R模型：**

**定义：世界是由一组称作实体的基本对象和这些对象之间的联系构成的，基本概念如下：**

**（1）实体(Entity)：客观存在并可相互区分的事物叫实体，如学生张三、工人李四、计算机系、数据库概论。**

**（2）属性(Attribute)：实体所具有的某一特性，一个实体可以由若干个属性来刻画，例如，学生可由学号、姓名、年龄、系等组成**

**（3）域(Domain)：属性的取值范围，例如，性别的域为（男、女），月份的域为１到12的整数**

**（4）实体型(Entity Type)：实体名与其属性名集合共同构成实体型，例，学生（学号、姓名、年龄、性别、系、年级）；注意实体型与实体（值）之间的区别，后者是前者的一个特例，如(9808100，王平，21，男，计算机系，2)是一个实体**

**（5）实体集(Entity Set)：**

**同型实体的集合称为实体集，如全体学生**

**（6）联系(Relationship)：**

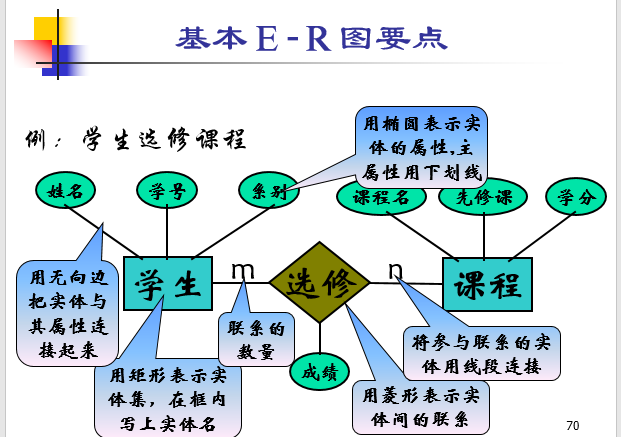
**实体之间的相互关联。如学生与老师间的授课关系，学生与学生间有班长关系**

**联系也可以有属性，如学生与课程之间有选课联系，每个选课联系都有一个成绩作为其属性，同类联系的集合称为联系集。**

**（7）元或度（Degree）：**

**参与联系的实体集的个数称为联系的元。如学生选修课程是二元联系，供应商向工程供应零件则是三元联系**

**3.E-R图绘制：**

****

* **（逻辑）数据模型：三要素、层次模型、网状模型、关系模型**

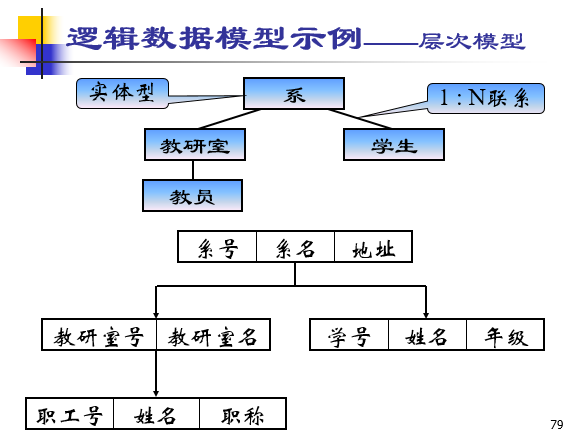
**1.三要素：数据结构、数据操作、数据的约束条件**

**2.层次模型：**

（1）用树结构表示实体之间联系的模型叫层次模型

（2）树由节点和连线组成，节点代表实体型，连线表示两实体型间的一对多联系

（3）结构简单，易于实现；但是支持的联系种类太少，只支持二元一对多联系；并且数据操纵不方便，如子结点的存取只能通过父结点来进行。

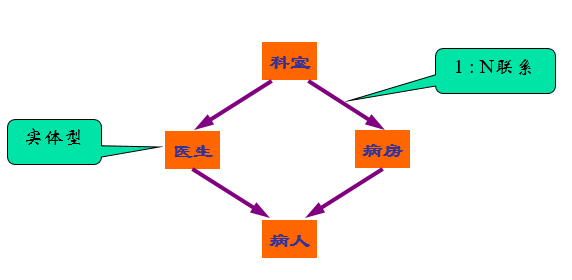


**3.网状模型：**

**（1）是一个满足下列条件的有向图：①可以有一个以上的节点无父节点 ②至少有一个节点有多于一个的父节点（排除树结构）；**

**（2）节点代表实体型，有向边（从箭尾到箭头）表示两实体型间的一对多联系**

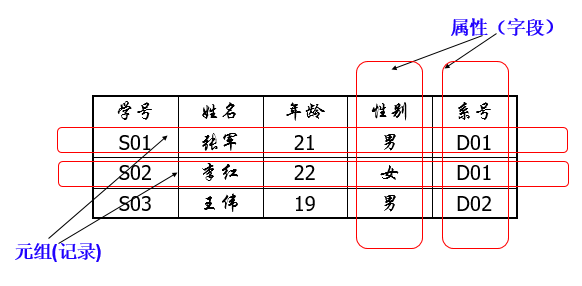
**（3）表达的联系种类丰富，性能良好，存取效率高；但是结构复杂，语言复杂。**

****

**4.关系模型：**

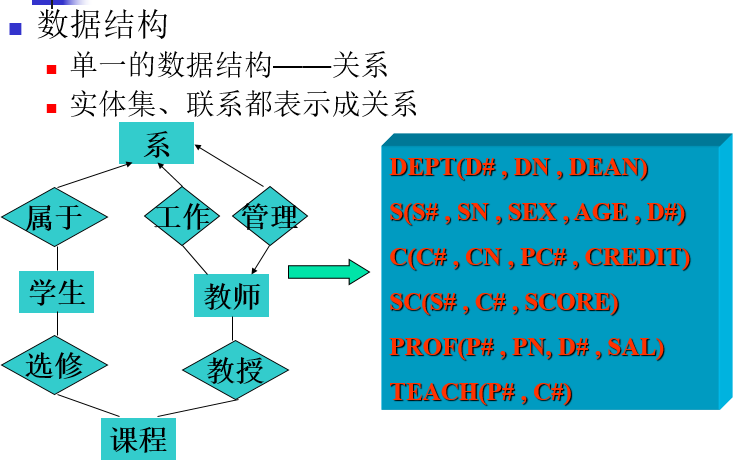
**（1）用二维表来表示实体及其相互联系**

**（2）简单，表的概念直观、单一，用户易理解**

****

第二章 关系模型与关系代数

常见关系模型：



* **关系的定义及性质**

**：笛卡尔积D1×D2×…×Dn的子集叫做在域D1 , D2 ,…, Dn上的关系，用R(D1 , D2 ,…, Dn )表示；关系是笛卡尔积中有意义的子集。**

* **关系模式与关系实例**

**1.** **列是同质的，一列中的分量来自同一域，是同一类型的数据，不同的列可来自同一域，每列必须有不同的属性名，行列的顺序无关紧要。**

**2. 任意两个元组不能完全相同（集合内不能有相同的两个元素）**

**3. 每一分量必须是不可再分的数据。满足这一条件的关系称作满足第一范式（1NF）的**

* **理解三类关系完整性**

**1. 实体完整性：主键非空非重复**

**2. 参照完整性（引用完整性）：外键值要么为对应主键的值，要么为空**

**3. 用户定义的完整性：用户针对具体的应用环境定义的完整性约束条件**

* **关系的候选键、主键、外键**

**1. 候选键：关系中的一个属性组，其值能唯一标识一个元组。若从属性组中去掉任何一个属性，它就不具有这一性质了，这样的属性组称作候选键/码**

**2. 主键/码（Primary Key）:**

**进行数据库设计时，从一个关系的多个候选键/码中选定一个作为主键/码**

**3. 外部键/码（Foreign Key）**

**关系R中的一个属性组，它不是R的键/码，但它与另一个关系S的键/码相对应，则称这个属性组为R的外部键/码,如S关系中的D#属性.**

**拓:外键可以是以下四种情况之一:①改名后的主码 ②主码之一 ③引用自己的主码 ④另一个关系的主码直接搬过来**

* **用关系代数表达查询**
* **关系代数的基本运算**

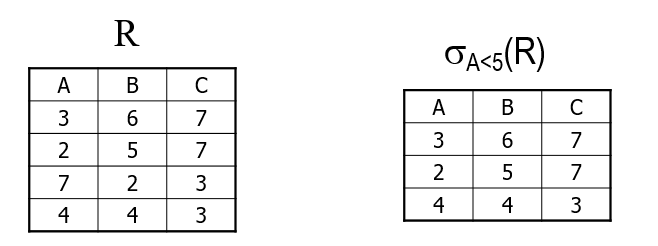
**包括一元运算(选择、投影、更名)与多元运算(笛卡儿积、并、集合差)**

* **掌握选择、投影、连接、差运算等**

**1.选择:** **在关系R中选择满足给定条件的元组（从行的角度）**

**(1)语法: σF(R)**

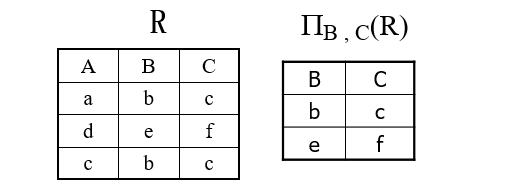
**(2)示例:**

****

**2.投影:** **从关系R中取若干列组成新的关系（从列的角度）**

**(1)语法:** II**ΠA(R)**

**(2)示例:**

****

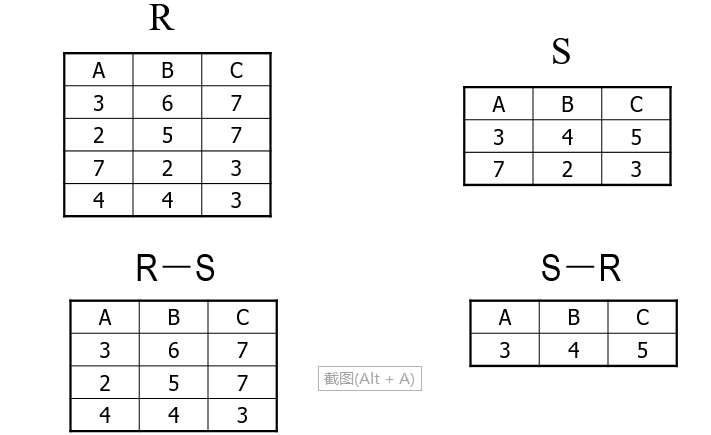
**3. 连接:** **将两张不同的表链接**

|  |  |
| --- | --- |
|  | **示例** |
| **广义笛卡尔积**  **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687838006540.png** | **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687837493293.png** |
| **θ连接(当θ为等号时为等值连接)**  **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687837758605.png**  **注:由于A与B可以不同,相同则类似不去重复的自然连接** | **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687837977969.png** |
| **自然连接**  **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687838084515.png**  **注:** **自然连接中相等的分量必须是相同的属性组，并且要在结果中去掉重复的属性，而等值连接则不必;** **当R与S无相同属性时，结果为广义笛卡尔积** | **C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687838241938.png** |

**4. 差运算: 选出出现在一个关系而不在另一关系中的元组集合(两个关系必须是相容的,即属性数目相同,位置对应的属性的域必须相同)**

**(1)语法: R−S**

**(2)示例:**

****

* **除运算不考查**
* **关系演算不考查**
* **查询优化原则(补充)**

**①尽早进行选择运算与投影运算**

**②避免进行笛卡尔积运算**

**第三章 SQL语言**

* **区别DDL、DML所完成功能的差异**

**1.** **DDL语言（Data-Definition Language）：数据定义语言**

**①描述外模式、模式、内模式（源模式）**

**②模式翻译程序，把源模式翻译成目标模式，存入数据字典中**

**注:在SQL中对应create, alter, drop 等**

**2.** **DML语言（Data-manipulation language）:数据操纵语言**

**对数据库进行检索、插入、修改、删除；分为宿主型（预编译和增强编译）和自含型（解释执行）。**

**注:在SQL对应**[**insert**](https://so.csdn.net/so/search?q=insert&spm=1001.2101.3001.7020)**, delete, update 等**

* **掌握SQL的各种数据定义语句（表及完整性、视图）**

|  |  |
| --- | --- |
| 表与完整性 | C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687839060216.png |
| 视图 | C:\Users\Administrator\Documents\WeChat Files\wxid_swruuax8o8zt22\FileStorage\Temp\1687839462884.png |

* **SQL语言表达查询(结合作业复习)**
* **查询语句中各子句执行的逻辑顺序**

**from**

**join**

**on**

**where**

**group by(开始使用select中的别名，后面的语句中都可以使用)**

**avg, sum…**

**having**

**select**

**distinct**

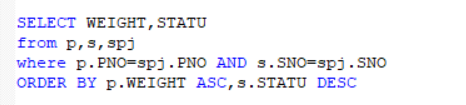
**order by**

**limit**

* **ORDER BY子句相关知识（多个属性、空值、升序降序等）**

**1.语法 : order by 列名 [asc | desc]**

**2.示例(多个属性排序):**

****

**注:如果需要对空值进行要求(排在最前或最后,可以参考如下例子)**

**强制把sort为空值的记录排在后面**

**select id,sort from fwzl\_house**

**where delFlag = 0 and id in (9807,9786,9638,9679)**

**order by if(isnull(sort),1,0) , sort asc**

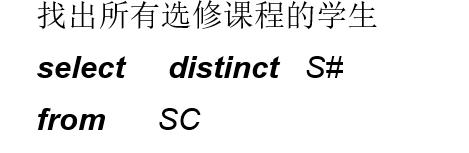
**注:if(条件,true时的返回值,false时的返回值)**

* **去重复**

**：select后面添加【语法约束】**

**1. 语法约束缺省为保留重复元组，也可用关键字all显式指明。若要去掉重复元组，可用关键字distinct或unique指明**

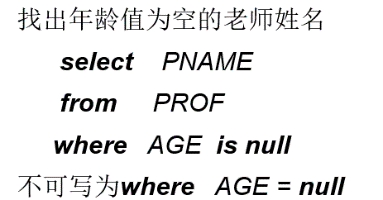
**2. 示例：**



* **空值处理（is null，除此以外不满足任何匹配条件）**

**1.语法: is [not] null**

**2.示例:**

****

**注:** **如果null参与算术运算，则该算术表达式的值为null;**

**如果null参与比较运算，则结果可视为false。在SQL-92中可看成unknown;**

**如果null参与聚集运算，则除count(\*)之外其它聚集函数都忽略null(例如,sum与avg都会将运算范围缩小为非null的其他行)**

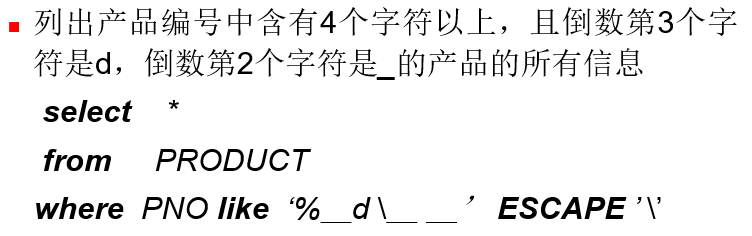
* **模糊查询（通配符的使用:%和\_）**

**1.语法:**

**列名 [not] like ‘字符串’**

**注:字符串中使用** **“%” 匹配零个或多个字符;使用“＿”匹配任意单个字符;末尾附加Escape定义转义字符**

**2.示例:**

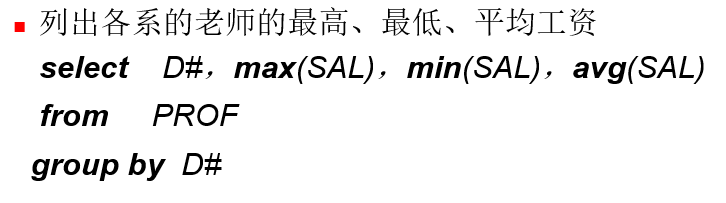
****

* **掌握查询的基本结构、分组聚集运算、分组的选择、不相关子查询（IN或NOT IN）等**

**1.分组命令:**

**(1)语法:** **group by 列名 [having 条件表达式]**

**(2)示例:**

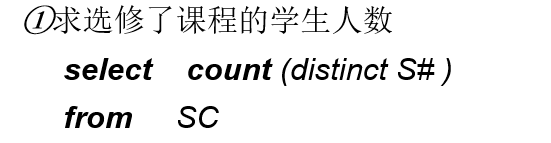
****

**2.聚集函数:**

**(1)常见函数: 平均值avg; 最小值min; 最大值max; 总和sum; 记数count**

**注:count()默认不数空值,如果count(\*)就会对空值计数;带distinct去重复去空值**

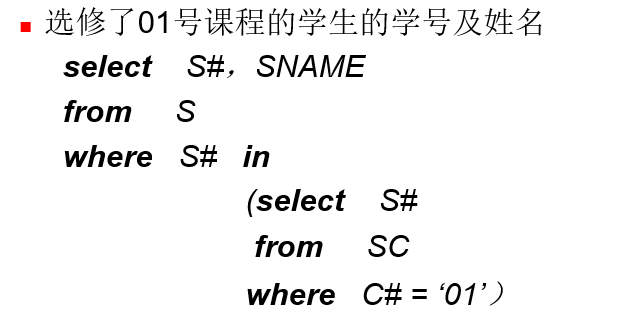
**(2)示例:**

****

**3.不相关子查询:**

**(1)语法: 表达式 [not] in （子查询）**

**(2)示例:**

****

* **带存在量词的相关子查询不考查**
* **SQL数据操纵语言**

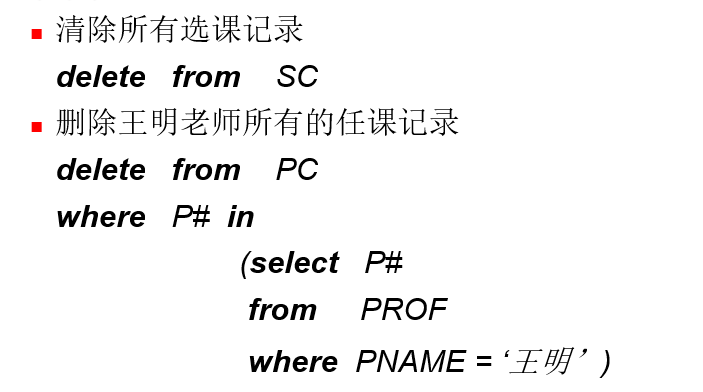
1. **insert 插入**

|  |  |  |
| --- | --- | --- |
|  | **语法** | **示例** |
| **插入一条元组** | **insert into 表名 [（列名[，列名]…]**  **values （值 [，值]…）** |  |
| **插入子查询结果中的若干条元组** | **insert into 表名 [（列名[，列名]…]**  **（子查询）** |  |

1. **delete删除操作**
2. **语法:delete from 表名 [where 条件表达式]**

**注:没有where默认删除所有元组**

1. **示例:**

****

1. **update更新操作:**
2. **语法:**

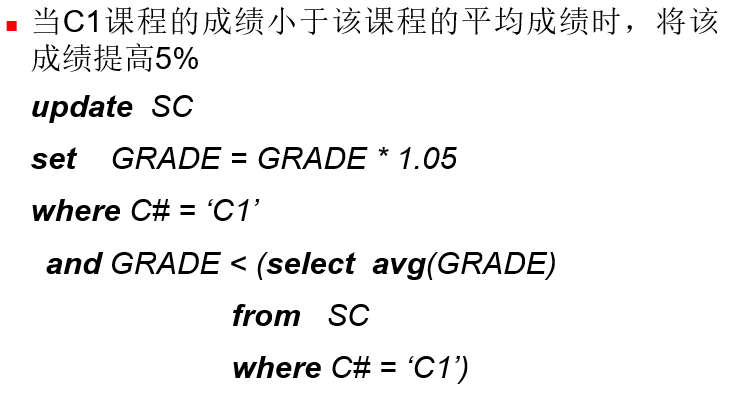
**update 表名**

**set 列名 = 表达式 | 子查询**

**列名 = [，表达式 | 子查询]…**

**[where 条件表达式]**

1. **示例:**

****

* **理解视图的本质和特点(命名的查询语句，虚拟关系；动态变化；行列子集视图（更新限制）)**

**1.本质:视图是命名的、从基本表中导出的虚表，它在物理上并不存在，存在的只是它的定义.因此它并不能提高查询效率**

**2.动态变化:即修改视图后可以更新基本表,但是需要定义行列子集视图(这种视图仅通过选择、投影操作导出，并且还包含了基本表的主码)**

1. **更新限制**

**①select子句中的目标列不能包含聚集函数**

**②select子句中不能使用unique或distinct关键字**

**③不能包括group by子句**

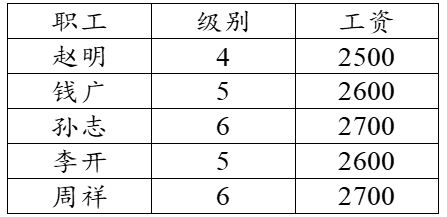
**④不能包括经算术表达式计算出来的列**

**⑤对于行列子集视图可以更新（视图是从单个基本表使用选择、投影操作导出 的，并且包含了基本表的主码）**

* **视图的优点:视图的优点:个性化服务;安全性;简化用户操作;逻辑独立性(基本表与外模式之间的映像)**

1. **关系数据模式设计**

* **关系模式设计中插入异常、删除异常、修改异常的具体含义、例子**

****

**注:级别确定唯一工资,两者捆绑在一起.**

|  |  |
| --- | --- |
|  | **例子** |
| **插入异常** | **如果没有职工具有8级工资，则8级工资的工资数额就难以插入** |
| **删除异常** | **如果仅有职工赵明具有4级工资，如果将赵明删除，则有关4级工资的工资数额信息也随之删除了** |
| **修改异常** | **如果将5级工资的工资数额调为2620，则需要找到每个具有5级工资的职工，逐一修改** |
| **数据冗余(补充)** | **职工很多，工资级别有限，每一级别的工资数额反复存储多次** |

* **函数依赖（平凡的、完全的）**

**1.定义:**

**设R(U)是属性集U上的关系模式，X , Y ⊆ U， r是R(U) 上的任意一个关系，**

**如果成立**

**对∀t , s ∈ r，若t[X] = s[X]，则t[Y] = s[Y]**

**那么称“X函数决定Y”，或“Y函数依赖于X”，记作X→Y**

**称X为决定因素.如S# → SN， （S#，C#）→ G**

1. **分类**

|  |  |  |
| --- | --- | --- |
| **平凡函数依赖** |  | **非平凡的:**  **右边的属性集不是左边的属性集的子集** |
| **部分函数依赖** |  | **部分的:**  **左边属性集的部分属性就能确定右边的属性集** |
| **传递函数依赖**  **(补充)** |  | **传递的:**  **三个属性集之间存在传递关系,并且属性集2既不决定属性集1也不包含属性集3** |

* **属性集闭包的计算、会利用来求关系模式的候选键(结合作业来复习,以下仅简单介绍思路)**

1. **求闭包:将选中的属性集放入F中不断迭代至属性集为U或不再变化**
2. **候选键:首先确定F中有哪些属性是不在右边(不在右边只能自己决定自己,因此一定在主属性中),将这些属性放入初始属性集;然后从一维开始不断增加维度迭代,每一维都要迭代至属性集为U;在迭代中,高维的属性集不能包含低维.**

* **理解与掌握1NF、2NF、3NF、BCNF，并能够进行判定。（若干结论：单属性主键、全是主属性、无非平凡函数依赖或全码的模式等）**

|  |  |  |
| --- | --- | --- |
|  | **定义** | **简单判定思路(反证法)** |
| **1NF** |  | **看属性值有没有集合或者序列,有就不是1NF** |
| **2NF** |  | **1NF基础上,看有没有非主属性部分函数依赖于主码,存在则不是** |
| **3NF** |  | **2NF基础上,看有没有非主属性传递函数依赖于主码,存在则不是** |
| **BCNF** |  | **3NF基础上,先找出所有非平凡函数依赖,**  **然后逐个判断左边是否含有主码,如果有一个不成立,则不是** |

**其他重要结论:**

**(1)候选码是单属性,一定可以达到2NF(部分函数依赖左边要求为属性组,即两个属 性以上)**

**(2)一个全是主属性的关系模式一定达到3NF(没有非主属性)**

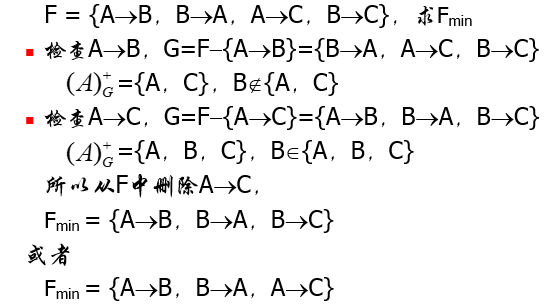
**(3)任何一个二目关系模式R(A,B)(构不成属性集,也就没有属性集间的包含关系,也就没有非平凡函数依赖)和一个全码的关系模式一定可以达到BCNF(没有非主属性)**

* **求解函数依赖集F的最小覆盖Fmin(补充)**

**1.定义:函数依赖集满足单属性化(右边全部单属性),无冗余化(去掉多余的依赖),既 约化(左边尽量少属性)**

|  |  |
| --- | --- |
| **算法** | **简单思路概括** |
|  | **①单属性化:将右边有k个属性的关系拆成k个右边只有一个的关系**  **②无冗余化:逐个拿出F中的关系,看拿出关系的左边关于剩下关系集的闭包是否包含右边,包含就去掉该关系**  **③既约化:观察左边是多个属性的依赖,对于每一个这样的依赖,逐个拿掉左边的属性,每次拿掉一个属性,看剩下的属性的闭包是否包含该依赖的右边,包含则去掉** |

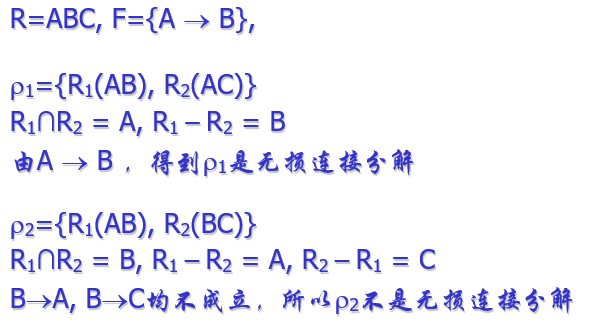
**2.示例:**

****

* **判断分解的无损连接性和保持函数依赖（包括判定算法和定理）**

1. **判断分解的无损连接性:**
2. **定理判断:仅适用于分解成两个范式,R1∩R2→R1−R2(或者U1∩U2→U1−U2)**

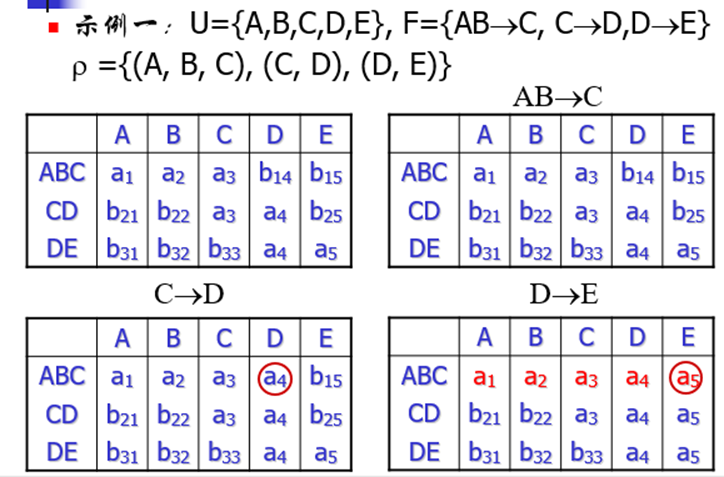
**示例:**

****

1. **矩阵判断(通法):**

|  |  |
| --- | --- |
| **算法** | **简单思路描述** |
|  | 1. **建立矩阵:行标签为各个分解模式的U分,列标签为U总的各个属性;逐行填写,已知标aj,未知标bij** 2. **逐个检查F总的函数依赖,每一次函数依赖检查中接着扫描矩阵的多行,按照【左边相等右边也要相等】的原则,如果若干行中,存在左边相等右边不等的情况,将右边统一成较小者(同标a<b;不同标,比较行号)**   **注:每一次函数依赖的检查都是在上一次的基础上进行的**   1. **直到出现一行全a或连续两次无变化后终止** |

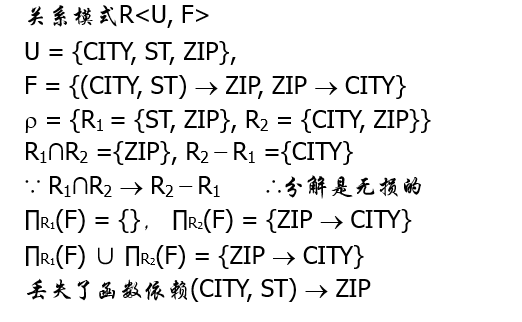
**示例:**

****

1. **保持函数依赖:**

**:正常来说,需要判断F = G,用闭包判断;但是考试中只要比较分解后的模式,分别求各个属性集在F上的闭包的并集,看并集是否缺失F中的函数依赖即可**

**示例:**

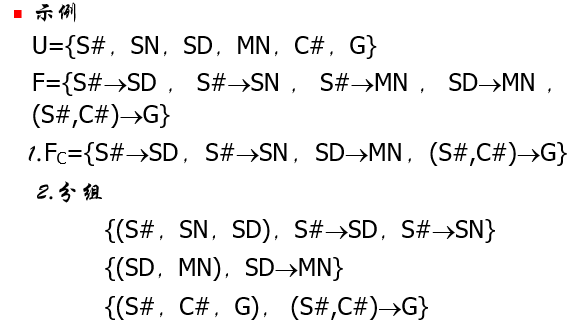
****

* **规范化到3NF，BCNF的相关方法**

1. **规范化到3NF:**

|  |  |
| --- | --- |
| **算法** | **简单思路描述** |
|  | 1. **求最小正则覆盖FC** 2. **如果FC中存在依赖的左右属性集并集是U,终止**   **3.分组:按照FC中函数依赖的相同左边来分组(模式),然后逐个判断这些分组是否达到3NF** |

**示例:**

****

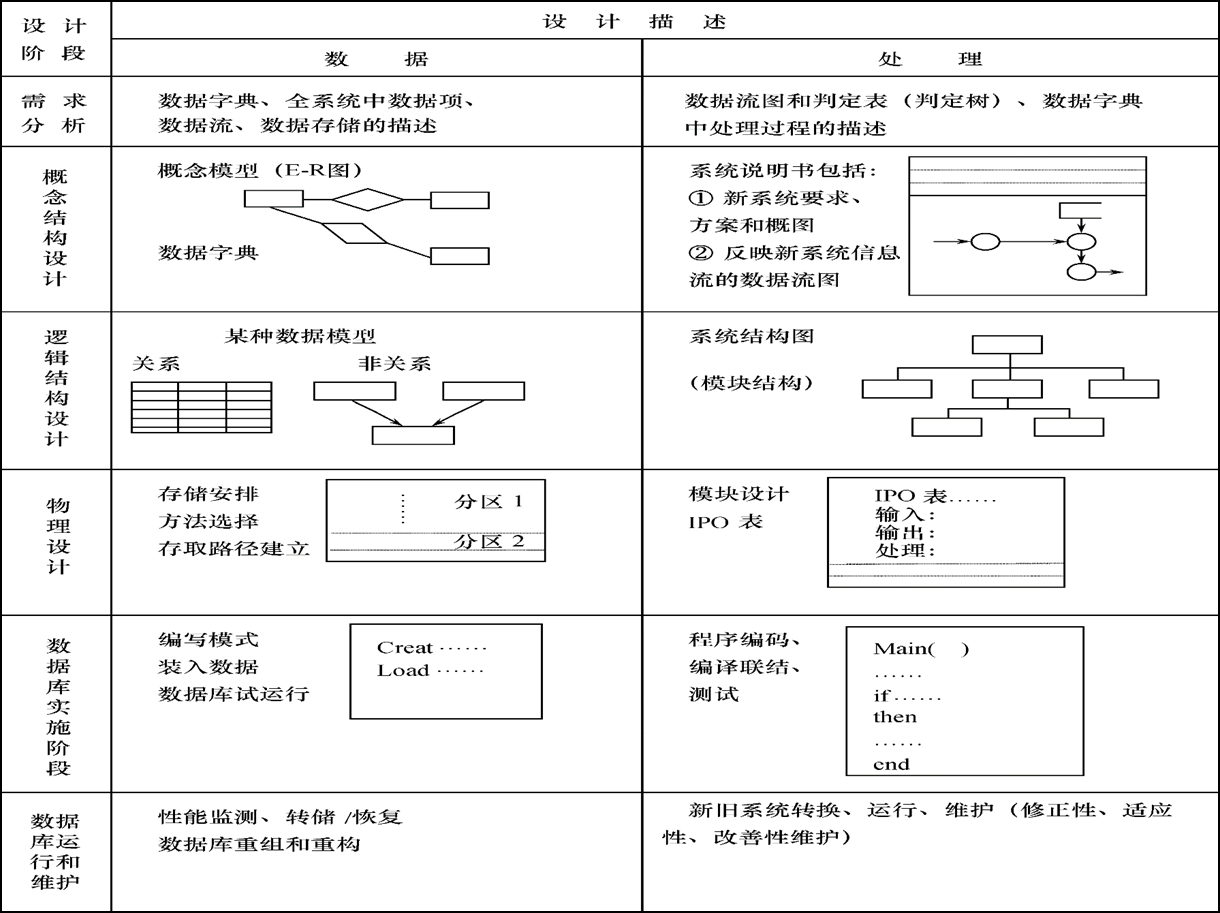
1. **规范化到BCNF:**

|  |  |
| --- | --- |
| **算法** | **简单思路描述** |
|  | 1. **观察是否都是BCNF** 2. **对于不是BCNF的模式,先求候选码;接着找非平凡函数依赖中,左边不是主码的依赖(这个关系也是破坏BCNF的原因,姑且称为""破坏依赖)** 3. **破坏BCNF的依赖关系拿出来单独作为一个模式,剩下的在属性中去掉破坏依赖的右边作为另外一个模式**   **4.迭代至全部都为BCNF** |

* **应用题型：给出关系和函数依赖**
  + **给出函数依赖（如果是具体例子）**
  + **求候选码**
  + **判断关系所属的最高范式，并给出证明。**
  + **将关系规范化到3NF，或者BCNF**
  + **判断或证明无损连接性**

**第五章 数据库设计**

* **数据库设计的各个阶段，每个阶段分别要做的主要工作**

****

* **ER图的各个组成部分(见上)**
* **会根据给定要求画ER图（EER模型不考查）(格式见上)**

1. **从最底层数据流图入手，分析其数据关系，绘制E-R图的草图**
2. **确定实体型及属性，再确定实体型之间是什么类型的联系，以及给该联系命名**

**示例:**

|  |  |
| --- | --- |
|  |  |
|  |  |

* **将ER图转化为关系模式**

1. **原则:实体集→关系, 实体属性→关系属性, 联系集→?**
2. **实体间联系的转换:**

|  |  |  |
| --- | --- | --- |
| **二元联系一对一联系** |  | **一个职工最多管理一个部门，一对一联系转换方式:**  **①将联系加到职工里，如将部门的码加入到职工属性中去(因此成为外码)**  **②将联系加到部门里面，同上。**  **③管理单独做一个关系，将两边的码加入管理联系**  **注:选②最好，每个部门都有主管，以后关联部门表和职工表连接即可完成查询;③需要三张表连接;①不是每个职工都是领导** |
| **二元联系一对多联系** |  | **将1一方实体型的主码，加到n一方实体型的属性集中;如将班主任加入到学生的属性中** |
| **二元联系多对多联系** |  | **需要直接转化成一个新的关系，双方关系的主码都加入作为新关系的主码:同一个实体内部的多对多属性，也是如此 (不过要注意重命名区分主码)** |
| **多元联系** |  | **联系自己与属性加各个实体集的主码** |

* **ER图中的联系与关系模式外键的关系(见上)**
* **能分析出关系模式的主键、外键(见上)**

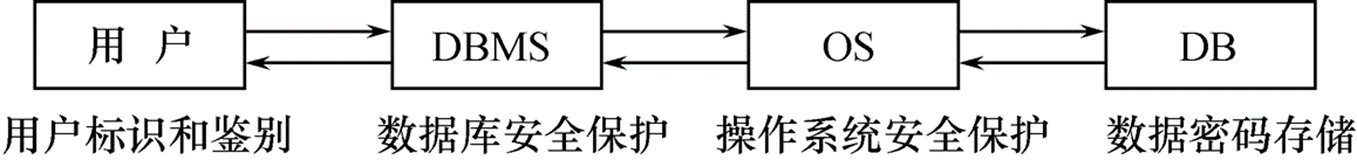
**第六章 安全性与完整性**

* **安全性与完整性差别**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **安全性** | **完整性** | |
| **目的** | **为了防止非法用户的非法访问** | **防止合法用户的错误操作，从而防止错误数据进入数据库** |  |
| **防范对象** | **非法用户与非法操作** | **不合语义的、不正确的数据** |  |

* **实现数据库安全性的措施**

**计算机不同层级的安全措施**

****

**数据库安全性保护措施:**

**基于视图技术、基于访问控制技术、审计追踪技术、其他安全技术（统计数据库，数 据加密）**

* **角色的定义及作用**

**角色是被命名的一组与数据库操作相关的权限。角色是批量用户授权操作的需要， 简化了授权的过程**

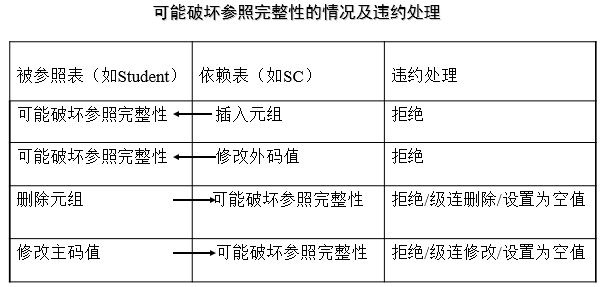
* **关系模型中的三类完整性规则，各有什么要求**

**1.实体完整性：唯一非空非重复**

**2.参照完整性：参照要么是存在的主码，要么为空**

**3.域完整性：非空约束、属性检查子句，域约束子句**

* **针对基本关系或依赖关系，哪些操作会破坏完整性；各种操作破坏这些完整性规则的处理方法（拒绝、级联、置空）**

****

* **触发器的定义及主要作用（详细语法结构不考查）**

**触发器是在有触发事件后，满足一定的触发条件而执行一组动作的一个特殊的存储 过程。它主要用来实现完整性，也可以实现一些自动操作。**

**第七章 数据库事务管理**

* **事务的定义**

**：事务是由一系列操作序列构成的程序执行单元，这些操作要么都做，要么都不做， 是一个不可分割的工作单位**

* **事务的ACID性质（包括确切含义、保障机制）**

|  |  |
| --- | --- |
| **原子性** | **事务中的所有操作要么都做要么都不做←恢复机制** |
| **一致性** | **事务的隔离执行必须保持数据库的一致性。事务开始前与结束后都处于一致性，这个一致性状态由用户负责←并发控制机制** |
| **隔离性** | **系统必须保证事务不受其他并发执行事务的影响←并发控制机制** |
| **持久性** | **一个事务一旦提交，它对数据库的影响必须是永久的；系统发生故障不能改变事务的持久性 ←恢复机制** |

* **数据库故障恢复的实现技术（数据转储、登记日志文件）**

1. **数据转储：由DBA定期将DB复制得到后备副本保存起来，分为静态与动态，或是海量与增量**
2. **登记日志文件：和后援副本综合起来回复数据库**

* **登记日志的原则（“先写日志文件原则”）**

1. **登记次序严格按并发事务执行的时间次序**
2. **必须先写日志文件，后写数据库**

* **日志文件记录的内容**

|  |  |
| --- | --- |
| **以记录为单位的日志文件** | **事务的开始与结束标记以及各个事务的所有更新操作** |
| **以数据块为单位的日志文件** | **事务标识和更新的数据块** |

* **数据库中几种类型的故障；故障的起因（死锁属于事务故障）；恢复策略；恢复过程由谁完成（用户干预？）**

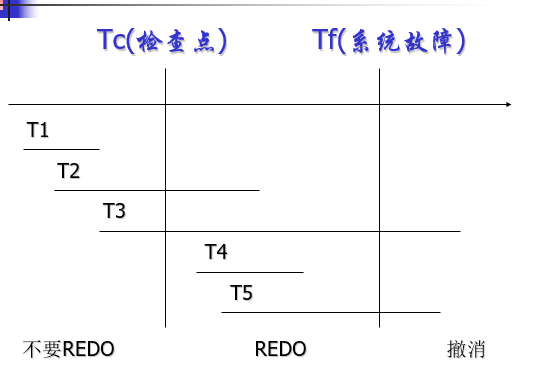
|  |  |  |  |
| --- | --- | --- | --- |
|  | **故障起因** | **恢复策略** | **恢复过程负责** |
| **事务内部故障** | **可预期的（余额不足）；非预期的（运算溢出、并发事务死锁、违反了某些完整性限制等）** | **通过事务撤消（UNDO）恢复** | **系统自动完成，对用户透明** |
| **系统故障** | **死机、CPU故障、OS故障、突然停电等导致系统需要重新启动** | **UNDO未完成事务、REDO已提交事务** | **系统重启时自动完成，不需要用户干预** |
| **介质故障** | **硬故障（磁盘损坏等外存故障）** |  | **DBA完成** |
| **计算机病毒** |  |  |  |

* **检查点的主要作用。**

**：减少扫描整个日志文件与REDO处理的重新执行大量时间浪费，动态维护日志文 件与保存数据库状态**

* **系统故障带有检查点的恢复策略**

|  |  |
| --- | --- |
| **保存** |  |
| **恢复** |  |

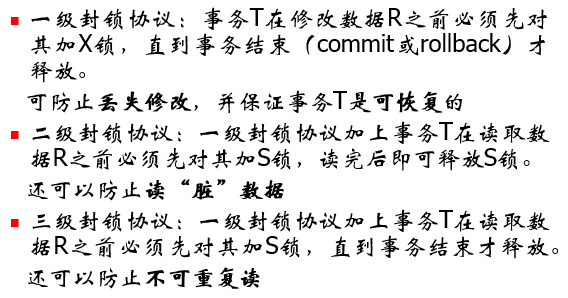
****

**注：undo可能undo到撤销之前去，但是redo不一定要从检查点之前开始**

* **数据库运行的基本单位：事务**
* **并发操作产生的三类数据不一致性问题，它们是如何发生的**

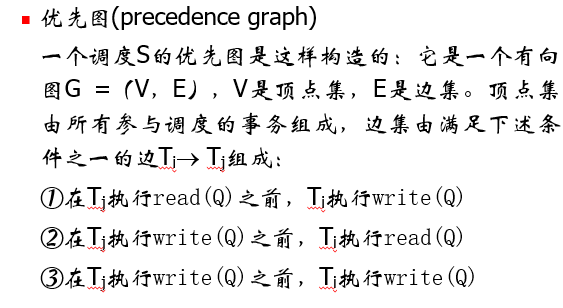
|  |  |
| --- | --- |
| **丢失修改** | **两个事务T1和T2读入同一数据并修改，T2提交的结果破坏了T1提交的结果，导致了T1的修改失效了** |
| **读“脏”数据** | **当事务T1修改某一数据时，事务T2读取同一数据后，T1由于某种原因撤销修改了，这时T1已修改过的数据恢复原值，而T2读到的数据是撤销修改之前的数据，那么此时T2的数据就与数据库中的数据不一致，则T2读到的数据就为"脏"数据，即不正确的数据** |
| **不可重复读** | **事务T1读取数据后，事务T2执行更新操作，使T1无法再现前一次读取结果** |

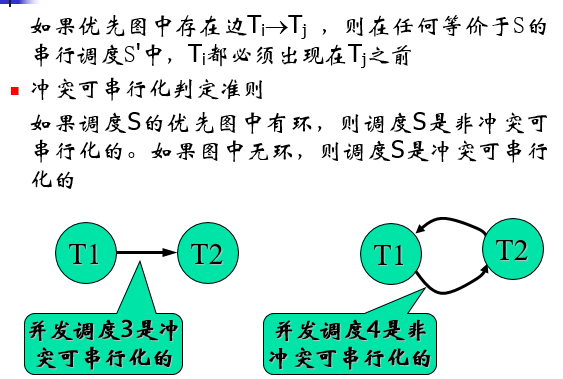
* **三级封锁协议具体内容及分别解决哪些不一致性问题**



* **并发调度的正确性准则及判断方法**

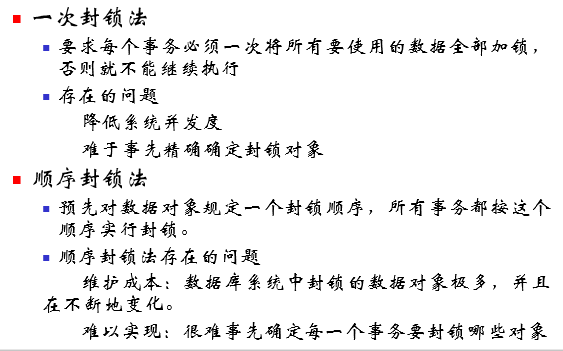
可串行性（Serializability)是并发事务正确性的准则

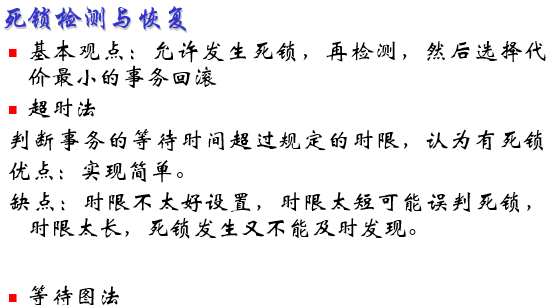


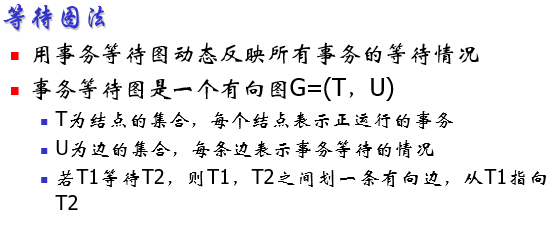


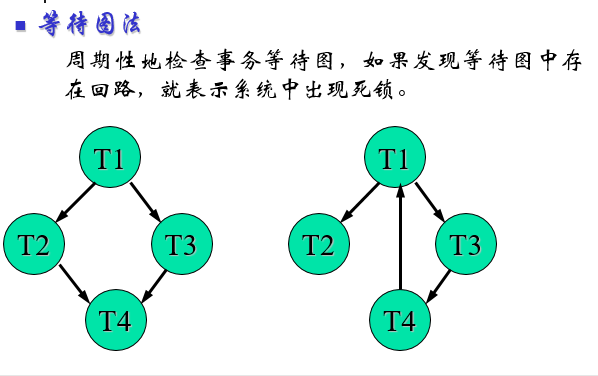
* **死锁的概念及解决方法**

1. 概念：如果有两个或两个以上事务同时要对两个或两个以上不同的数据加锁就可能发生相互等待的问题，从而形成死锁
2. 解决方法：

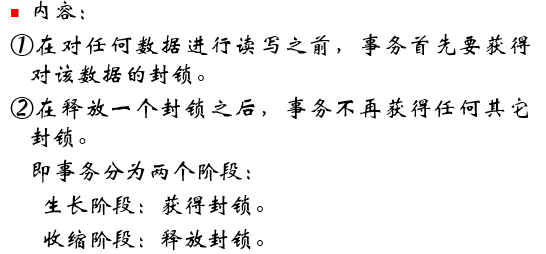








* **两段锁协议的内容。**



* + **定理：如果所有事务都遵守两段锁协议，调度一定是可串行化的；**
  + **两段锁协议不能防止死锁的发生**