数据库设计理论课整理

目录

[数据库设计理论课整理 1](#_Toc14078)

**[一、 数据库技术的起源与发展](#_Toc24578)** [2](#_Toc24578)

[1、 数据库管理技术的发展阶段 2](#_Toc13118)

[2、 数据库系统的基本概念 2](#_Toc27681)

[3、 数据库作为数据存储方法的特点 2](#_Toc4452)

[4、 数据库管理系统的结构演变 3](#_Toc15045)

**[二、 关系模型](#_Toc6235)** [3](#_Toc6235)

[1、 数据模型的基本概念、建模的要求 3](#_Toc17116)

[2、 数据模型的三级结构模式和两级映射 3](#_Toc18493)

[3、 关系模型的基本概念 4](#_Toc32093)

[4、 关系模型的优缺点 5](#_Toc21091)

[5、 关系代数 6](#_Toc27024)

**[三、 结构化查询语言SQL](#_Toc18174)** [8](#_Toc18174)

[1、 数据定义语言DDL 8](#_Toc21697)

[2、 数据查询语言SQL（略） 9](#_Toc19589)

[4、 高级SQL 9](#_Toc2174)

[5、 存储过程和触发器 11](#_Toc6007)

[6、 SQL嵌入方式（了解即可） 11](#_Toc23500)

**[四、 数据库设计](#_Toc19778)** [12](#_Toc19778)

[1、 数据库系统生存周期模型 12](#_Toc18164)

[2、 数据库概念设计与实体-联系图（ER图） 12](#_Toc4942)

[3、 关系模式规范化理论 13](#_Toc22239)

[4、 ER模型向关系模型的映射（只放总结，具体实例见ppt） 15](#_Toc21375)

**[五、 事务管理](#_Toc1044)** [16](#_Toc1044)

[1、 事务的概念和特性 16](#_Toc18860)

[2、 并发不加控制可能导致的问题 16](#_Toc21198)

[3、 事务的调度及其基本概念 17](#_Toc18904)

[4、 两段锁协议(Two-phase locking protocol) 18](#_Toc3286)

[5、 事务隔离等级含义，事务边界与隔离等级设计原则 20](#_Toc27958)

## 数据库技术的起源与发展

### 数据库管理技术的发展阶段

* 1. 手工管理：打孔卡片、磁带；无数据库管理软件；数据与程序相互依赖、无数据共享
  2. 文件管理：磁盘；OS下的文件系统；数据共享困难
  3. 数据库系统：大容量磁盘；专门的数据管理软件DBMS；数据实现彻底共享

### 数据库系统的基本概念

* 1. 数据：对客观对象的符号化表示 身高170cm
  2. 信息：数据+语义 我身高170cm
  3. **数据库( DataBase, 简称 DB ):** 数据库是长期储存在计算机内的､有组织的､可共享的、大量的数据的集合｡
* 数据的集合→有组织的→长期存储的、共享的、大量的
  1. **数据库管理系统( DataBase Management System, 简称DBMS ):** 数据库管理系统是位于用户与操作系统之间的一层数据管理软件,用于科学地组织和存储数据､高效地获取和维护数据。
* DBMS的主要功能包括数据定义功能､数据操纵功能､数据库的运行管理功能､数据库的建立和维护功能｡
* DBMS是数据库和用户程序之间的接口
* Oracle, DB2, SQLServer, MySQL, PostgreSQL……
  1. **数据库系统( DataBase System, 简称DBS )**: 数据库系统是由数据库､数据库管理系统(及其应用开发工具)､应用程序和数据库管理员组成的存储､管理､处理和维护数据的系统｡

### 数据库作为数据存储方法的特点

(1) 数据抽象

* 数据统一结构化

PS：缺点

需要购买DBMS数据库管理系统,有花费

需要满足一定的硬件设备

需要花时间和精力去学习使用和维护DBMS

* 结构信息独立由于数据

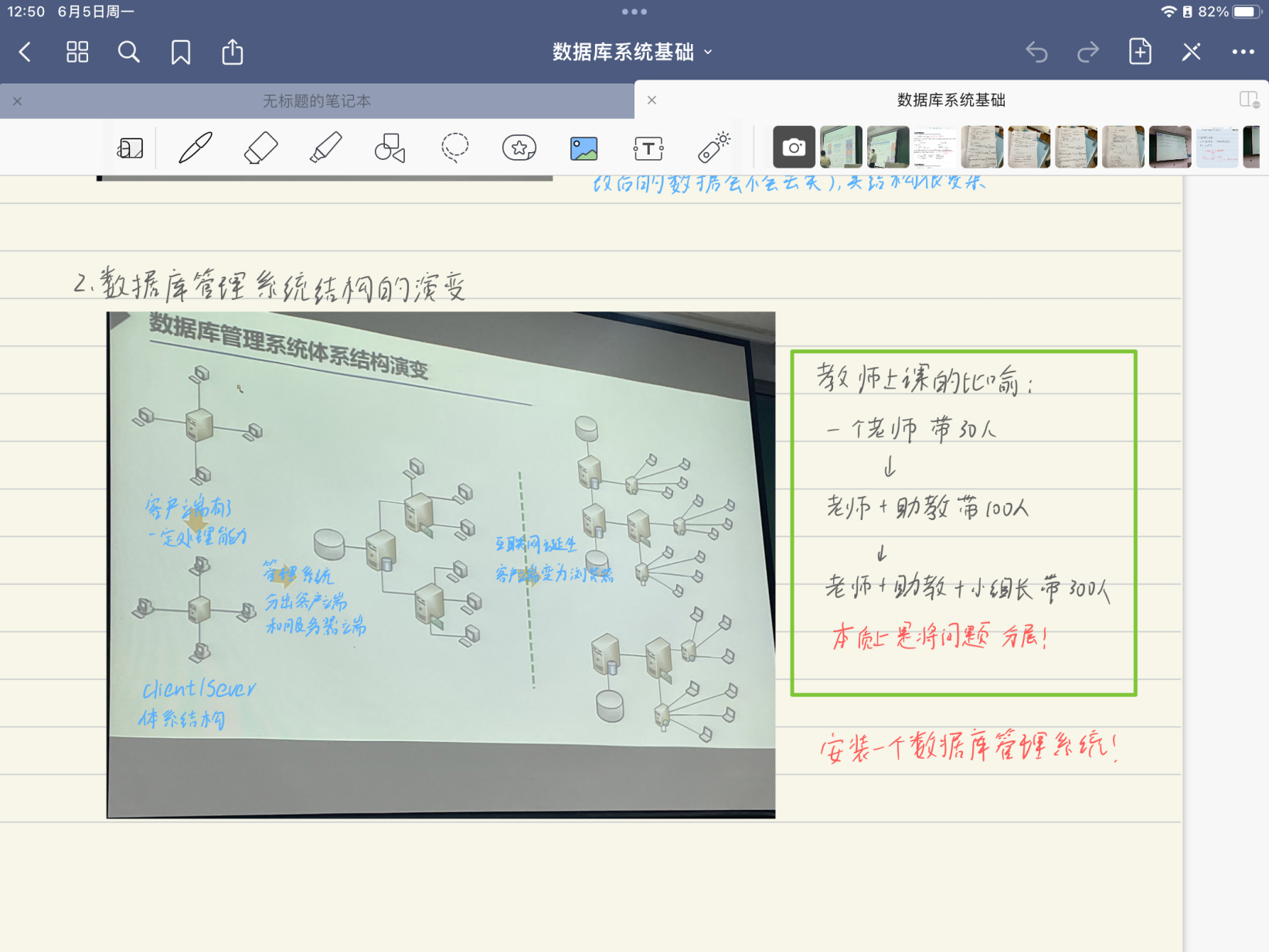
1. 高可靠

* 支持完整性约束
* 保证数据一致性

(3) 高效

* 物理存储优化
* 控制冗余
* 查询优化
* 并发访问控制

### 数据库管理系统的结构演变



## 关系模型

### 数据模型的基本概念、建模的要求

1. 基本概念

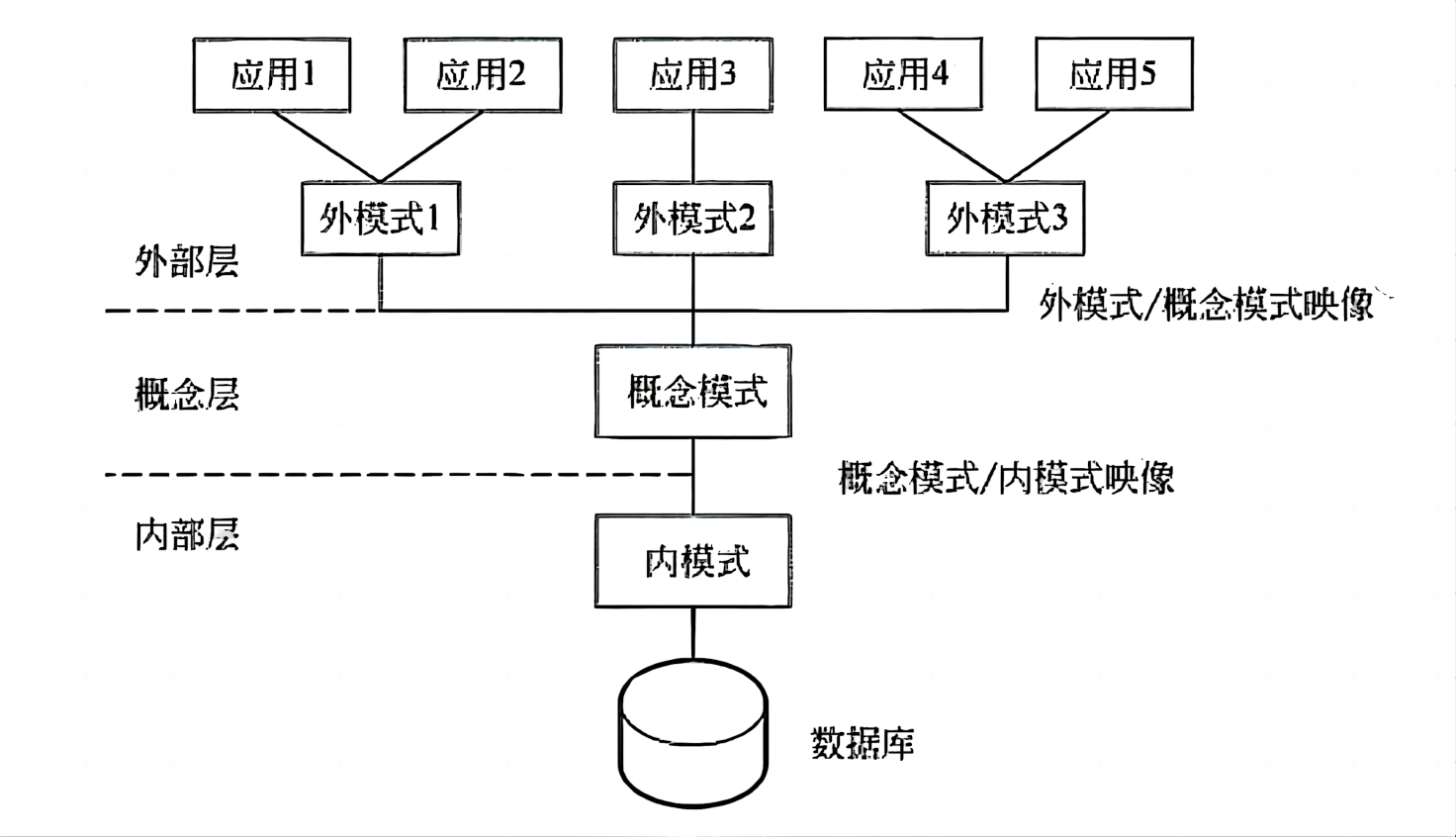
* 模型：对客观世界的模拟与抽象.
* 数据模型（作用）：对客观世界的数据特征的抽象，从抽象层次上描述了系统的静态特征、动态行为和约束条件，为数据库系统的信息表示与操作提供一个抽象的框架。

1. 数据模型建模的基本准则：反映客观世界、易理解、计算机容易实现
2. 数据模型三要素：

* 数据结构
* 数据操作
* 完整性约束（防止不规范的数据进入数据库）

### 数据模型的三级结构模式和两级映射

* **外模式**面向应用程序，描述用户的数据视图（View）
* **内模式（物理模式、存储模式）**面向物理上的数据库，描述数据在磁盘中如何存储
* **概念模式**（又称为模式、**逻辑模式**）面向数据库设计人员，描述数据的整体逻辑结构



* 外模式/概念模式映像体现了**逻辑独立性**。修改概念模式，不影响其上一层的外模式。
* 概念模式/内模式映像体现了**物理独立性**。修改内模式，不影响其上层的概念模式和外模式。

### 关系模型的基本概念

1. 关系模型的起源和定义

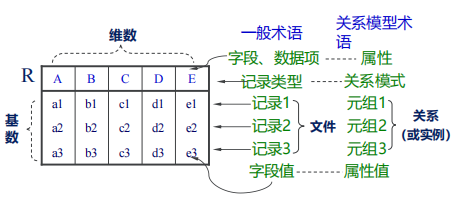


关系模型是指基于关系的数学概念、并且用二维表的形式表示实体和实体间联系的数据模型。

（用二维表格结构表示实体集,外键表示实体间联系的数据模型称为关系模型｡）

1. 关系模型的数据结构
   1. 关系中的特殊的码

* 超码(super key): 一个或多个属性的集合，可以唯一地标识一个实体
* 候选码(candidate key): 不含有多余属性的超码称为候选键（它的真子集不能唯一的标识一个元组）
* 主码(primary key): 用户选作元组标识的一个候选码称为主码
* 备选码/辅码(alternate key): 没有被选作主码的候选码
* 外码(foreign key): 外码引用另一个关系中的主键，用于建立两个实体之间的联系。



* 1. 关系模型的表示

Student (sNo, sName, sSex, sAge, sDept)

Course (cNo, cName, cPNo, cCredit)

SC (sNo, cNo, score)

1. 关系模型的完整性约束
   1. 关于空值Null

* 由于实际需要，引入Null，表示不确定或该属性不存在
* 任何数与Null比较，结果都是Null
* 引入Null后，逻辑体系由二值变成三值
  1. 实体完整性(Entity integrity)
* 每个关系中的主键值是唯一且非空的
  1. 参照完整性(Referential integrity)
* 外键值必须引用被引用关系的主键值，或者为NULL
* 例如表sutdent(有id,username,password),表sutdent\_info(有id,name,age,sex)｡其中表sutdent\_info参照了表student｡id作为外键｡那么**当student表删除一行时,表student\_info对应的id那一列将被删除或者置空。**同样,表student\_info增加一行,其中的id必须等于student表中的id
  1. 用户定义的完整性(Enterprise constraints)
* 用户自行定义的约束

### 关系模型的优缺点

1. 优点
2. 关系模型与非关系模型不同,它是建立在严格的数学概念的基础上的｡
3. 关系模型的概念单一，数据结构简单､清晰,用户易懂易用｡（无论实体还是实体之间的联系都用关系表示｡对数据的检索和更新结果也是关系(即表)）
4. 关系模型的存取路径对用户透明,从而具有更高的数据独立性､更好的安全保密性,也简化了程序员的工作和数据库开发建立的工作｡
5. 缺点
6. 对“现实世界”实体的表达能力比较弱
7. 语义过载
8. 不能很好的支持业务规则
9. 由于存取路径对用户透明,查询效率往往不如非关系数据模型｡因此,为了提高性能,必须对用户的查询请求进行优化,增加了开发数据库管理系统的难度

### 关系代数

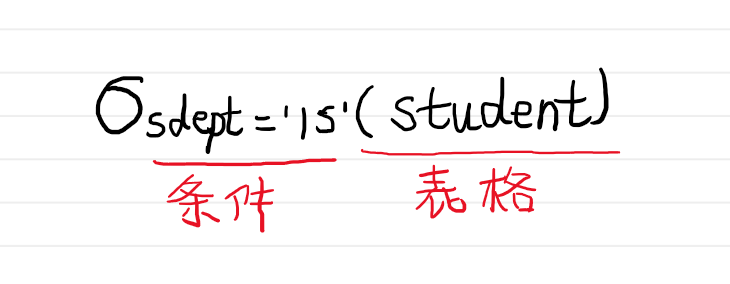
1. 并（Union）
   1. 表格列数一样
   2. 属性都一样
   3. 对重复的元组做去重处理
2. 差（Difference）
   1. 表格列数一样
   2. 属性都一样
3. 交（Intersection）
   1. 表格列数一样
   2. 属性都一样
4. 笛卡尔积（Cartesian product）
   1. 没有任何使用限制，万物皆可笛卡尔积
   2. R表：n元关系，k1个元组(k1行，n列)

S表：m元关系，k2个元组(k2行，m列)

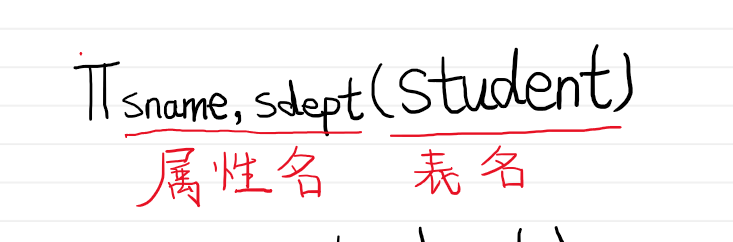
R表和S表进行笛卡尔积

得到一个(k1 \* k2)元，(n + m)列的新表

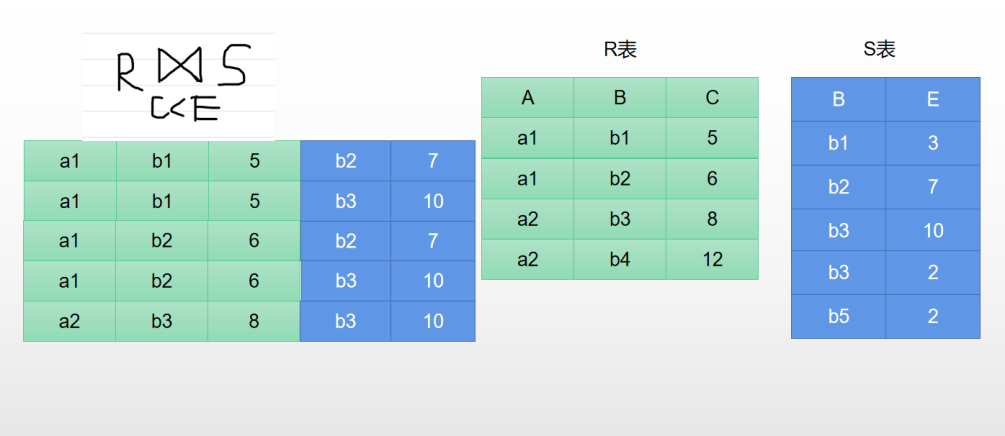
1. 选择（Selection）



1. 投影（Projection）



1. 连接（Join）
   1. 一般连接



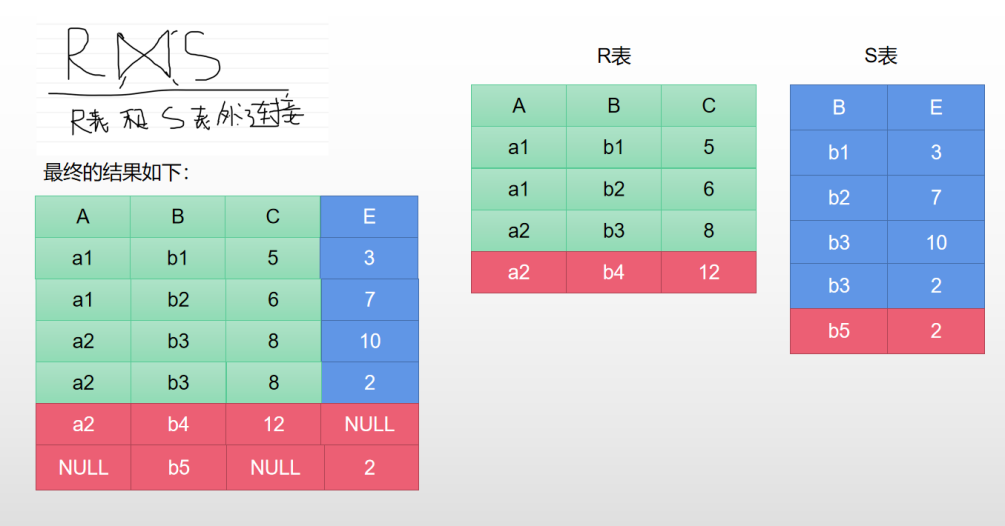
* 1. 相等连接



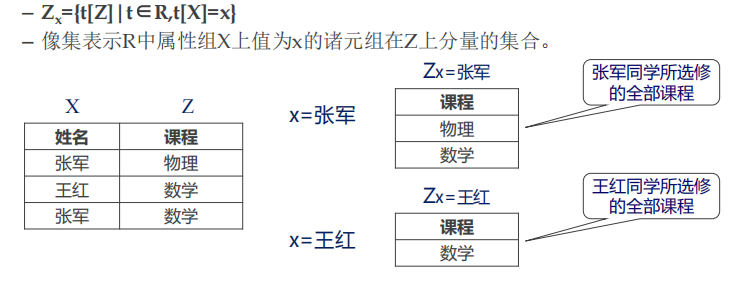
* 1. 自然连接



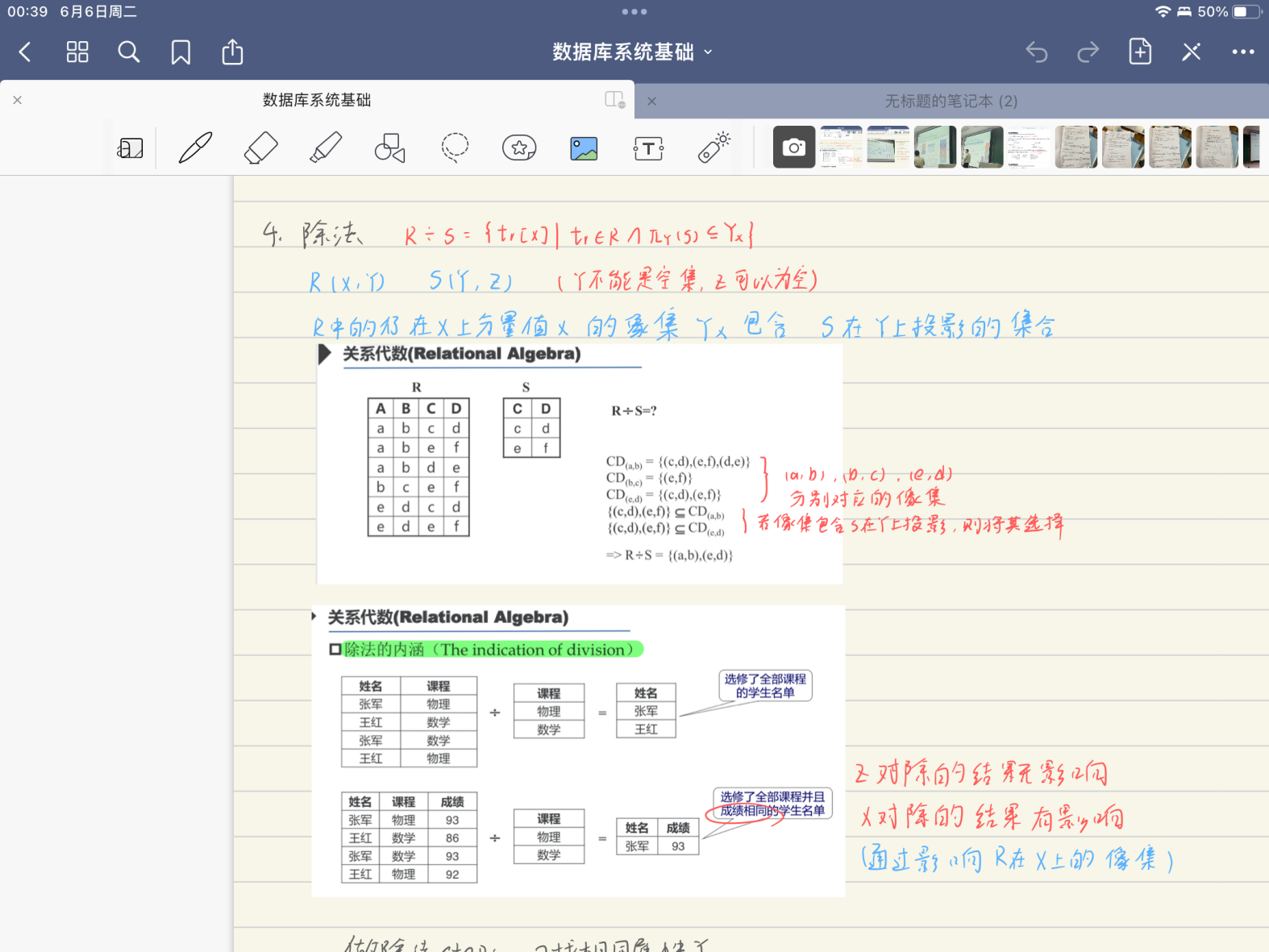
* 1. 外连接——解决连接后特有的元组被丢弃的问题



1. 像集（Images set）

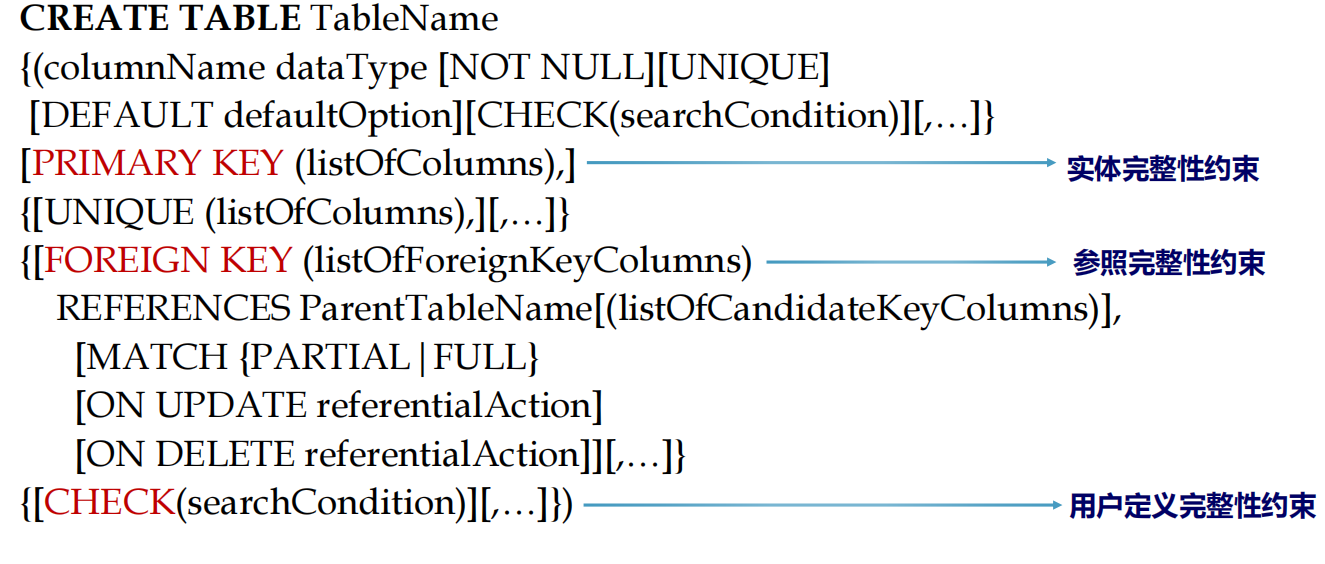


1. 除法（Division）

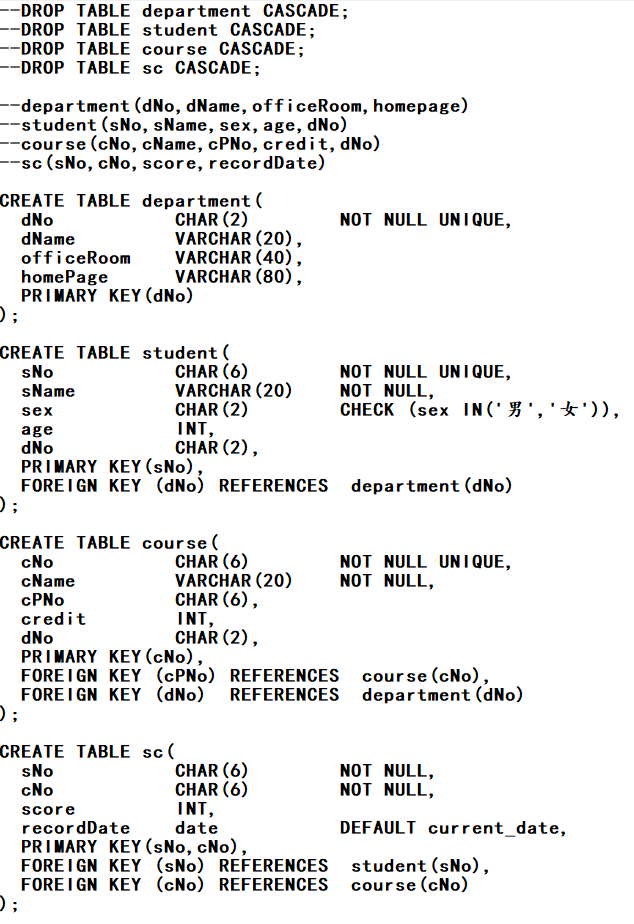


## 结构化查询语言SQL

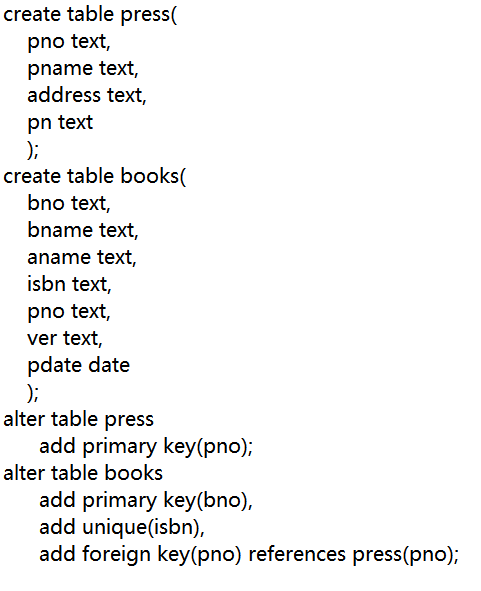
### 数据定义语言DDL



* 创建表的同时增加约束



* 在表创建之后增加约束



删除约束：

ALTER TABLE table\_name

DROP CONSTRAINT MyUniqueConstraint;

### 数据查询语言SQL（略）

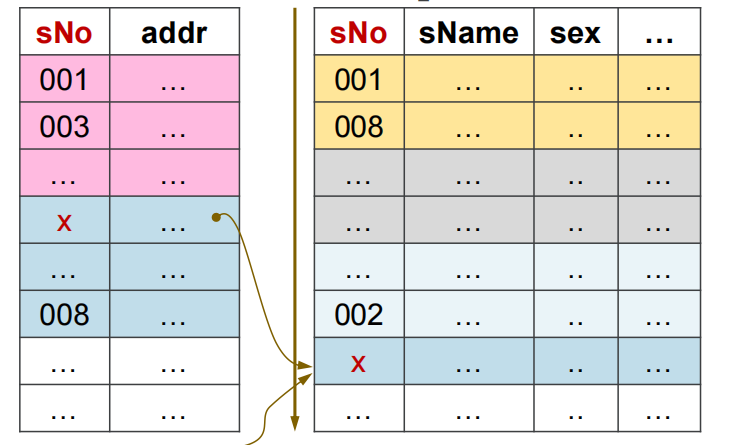
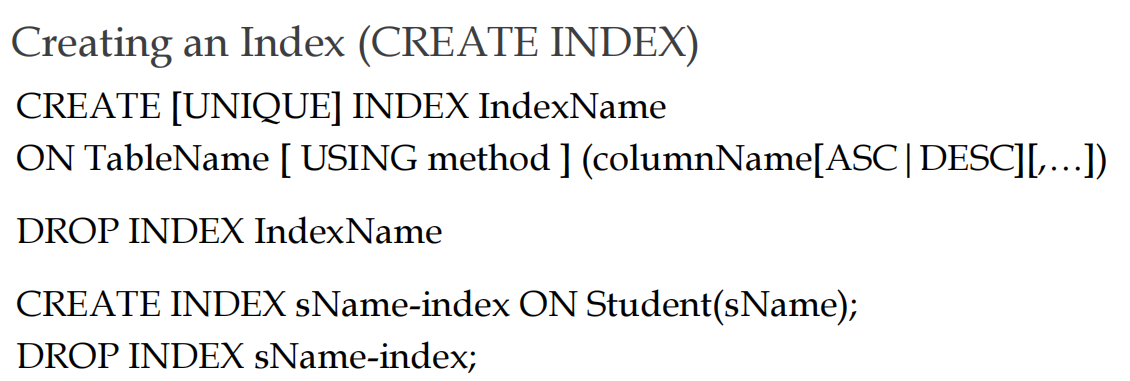
1. 数据操作语言DML（略）

### 高级SQL

1. 索引（Index）

**索引是一种基于一个或多个列的值提供加速对表的行的访问的结构。**

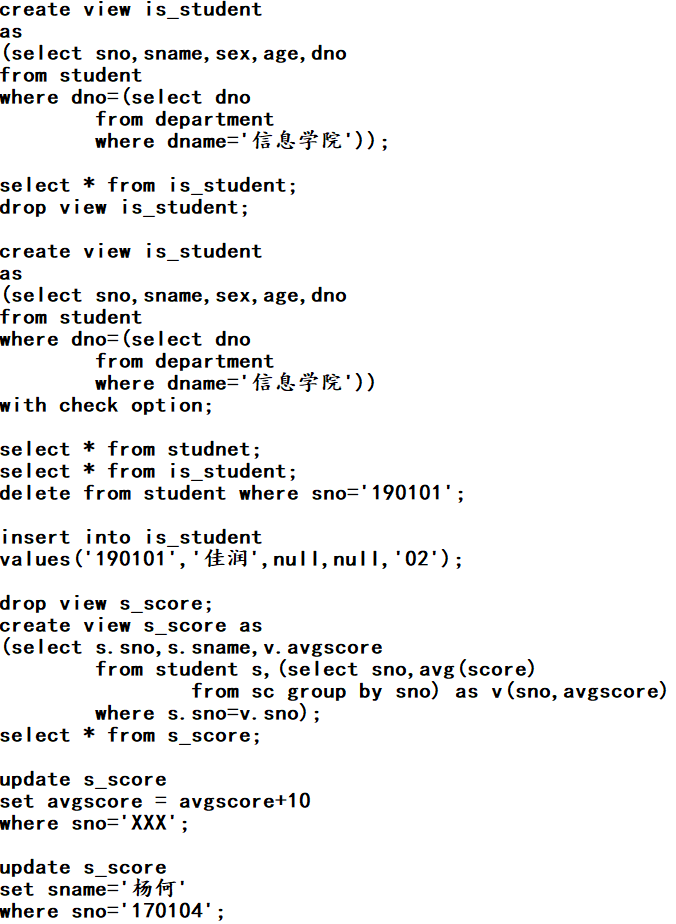
创建索引：



1. 视图（View）

**视图，是一个虚拟表，其内容由查询定义。视图本身不独立存储在数据库中，是一个虚表。即数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的基本表中。**

视图的优点：简化查询、安全性



视图数据的更新：

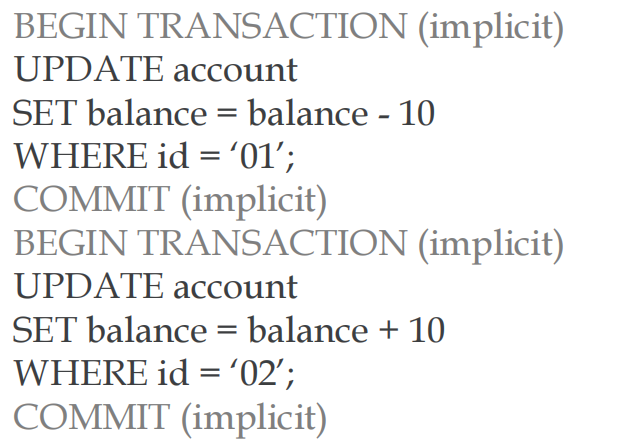
基本表→视图：基本表更新，视图随之更新

视图→基本表：视图的更新也可映射为基本表的更新。可以通过对视图执行INSERT、UPDATE和DELETE语句来直接更新视图的数据。下面考虑视图是否可更新：

WITH CHECK OPTION：使用INSERT或UPDATE语句更新视图时，必须满足与视图或子查询相关联的所有条件。如果不满足这些条件，数据库会拒绝该操作并引发错误。

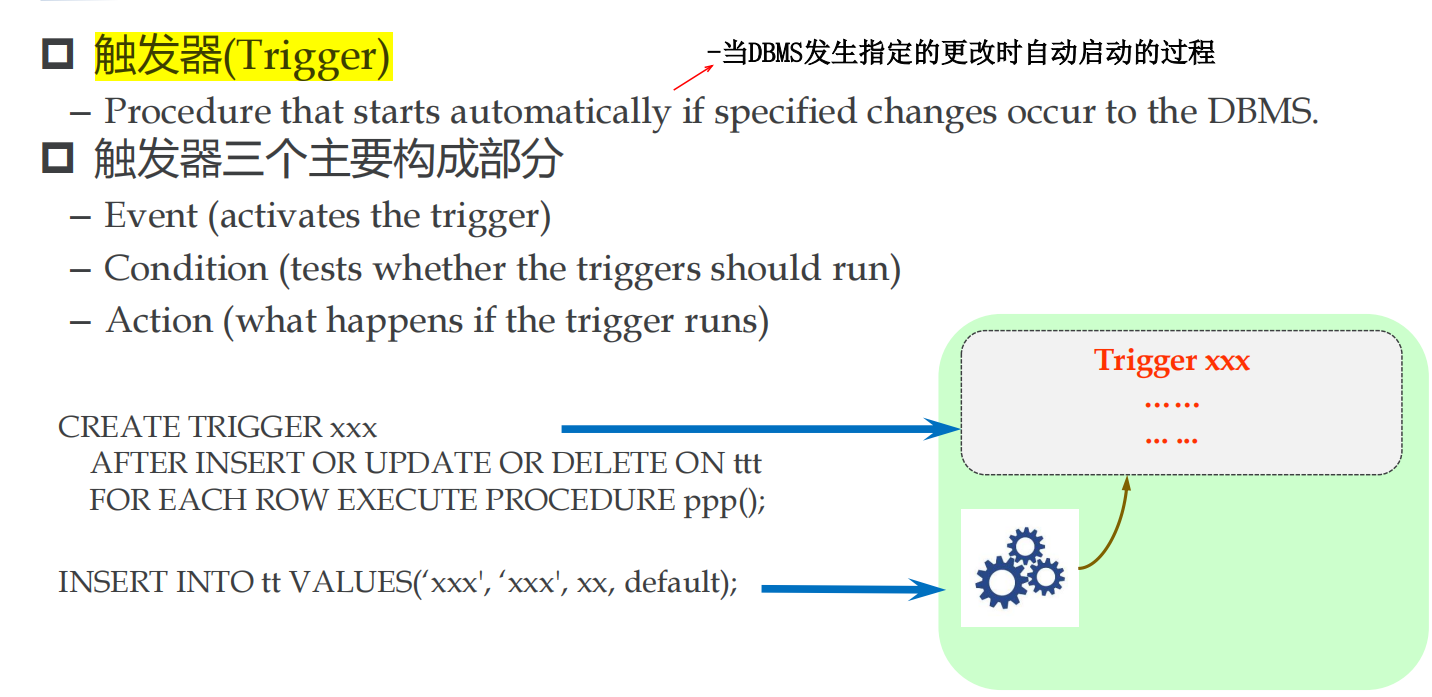
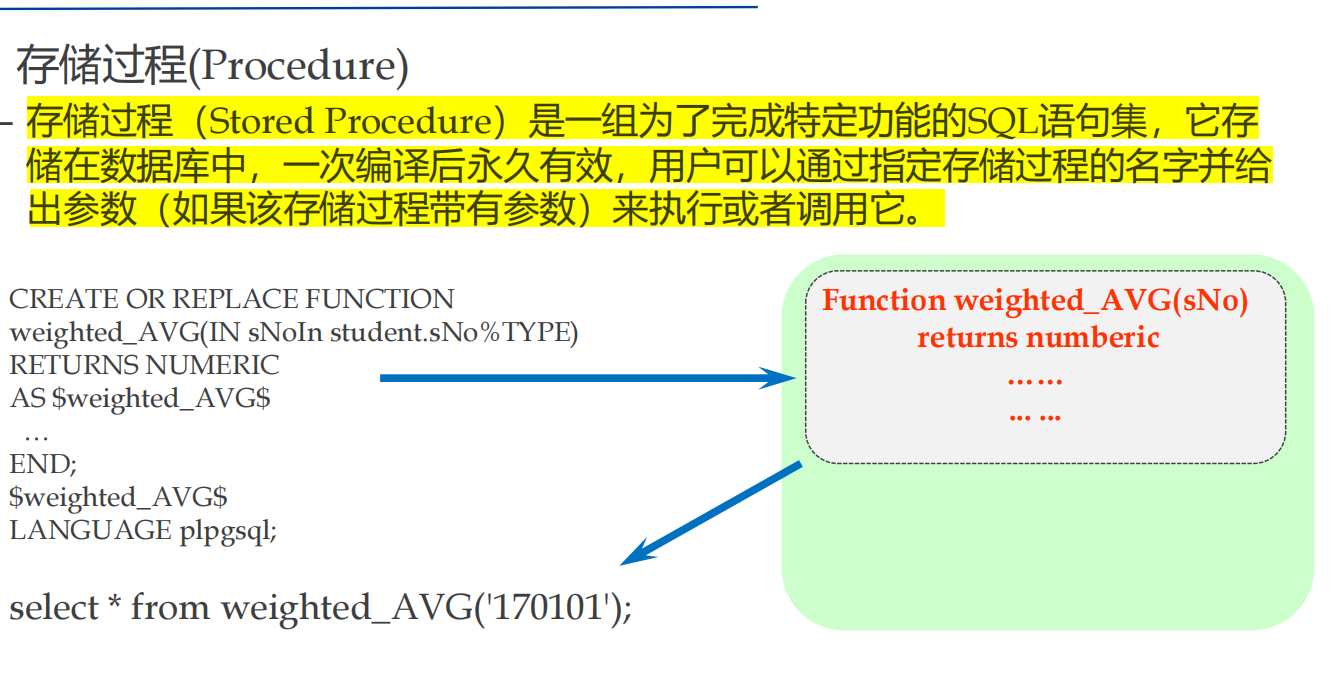
1. 事务（Transaction）

**事务是完成用户某一特定任务的与数据库的一次或多次交互的逻辑单位。**

****

1. 完整性约束（略）
2. 访问控制（略）

### 存储过程和触发器

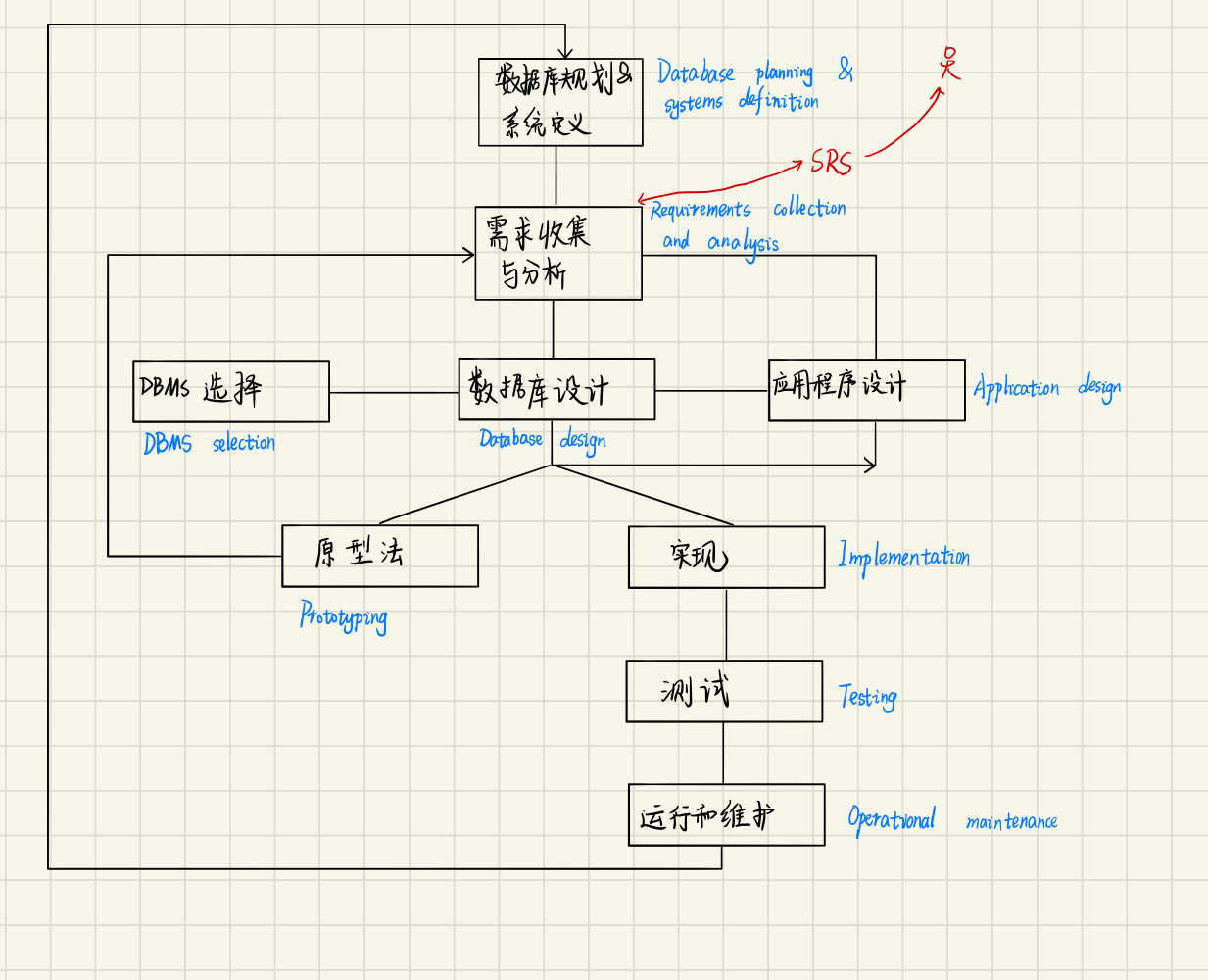


### SQL嵌入方式（了解即可）

阻抗失配、静态嵌入、动态嵌入、应用编程接口API、MVC

## 数据库设计

### 数据库系统生存周期模型



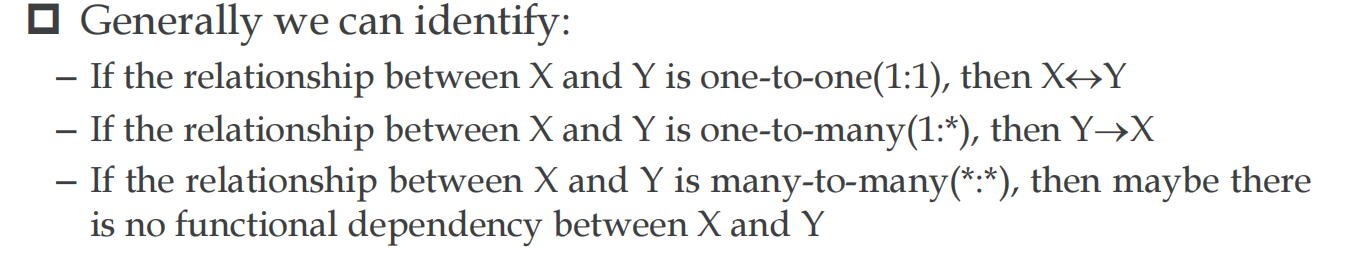
**数据库设计的三个阶段: 概念设计 逻辑设计 物理设计**

### 数据库概念设计与实体-联系图（ER图）

看ppt

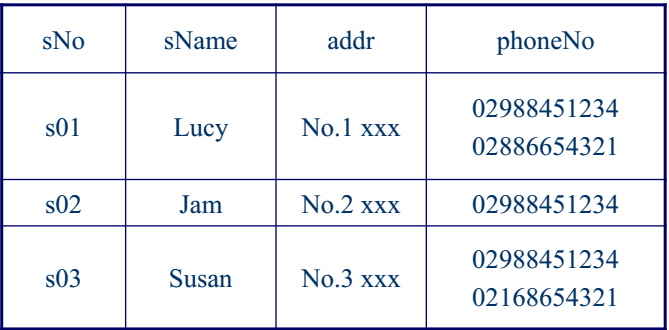
### 关系模式规范化理论

1. 函数依赖





1. 1NF
   1. 要求：**关系模式中属性不可分**



* 1. 问题：没法用表存储
  2. 解决办法：增加行、增加列、拆分成两个关系
* 增加行（通用）

优点：不受学生电话号码数量的限制

缺点：sNo的主码地位丢失；数据冗余

* 增加列

优点：sNo的主码地位不受影响；不会造成数据冗余

缺点：浪费存储空间

* 拆分成两个关系

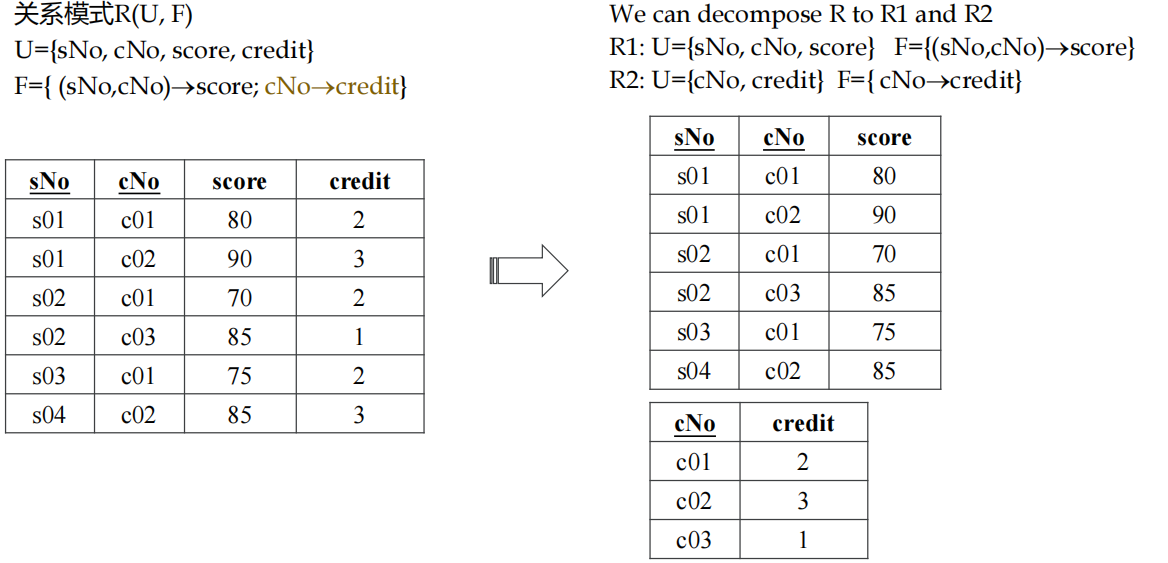
优点：消除冗余；不受学生电话号码数量的限制

缺点：后续查询需要连接，效率会降低

1. 2NF
   1. 要求：**关系模式中不能存在非主属性（不是候选码的属性）对码的部分依赖**

部分依赖是指一个非主属性，依赖于关系中的一部分而不是整个候选码。

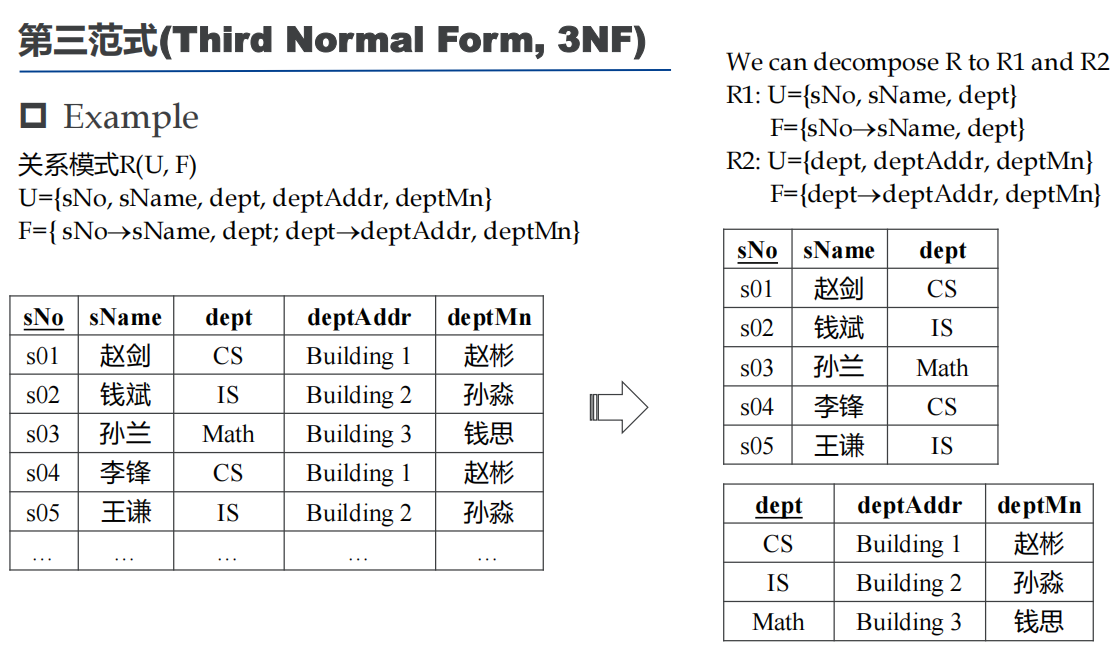
如果候选码是单个属性，则一定不会存在非主属性对码的部分依赖，则至少满足第二范式的要求。



* 1. 问题：增删改异常问题
  2. 解决办法：把部分依赖于码的属性连同它依赖的属性分离出去，组成一个新的关系模式。同时，在原来的关系模式中要保留一个外键。

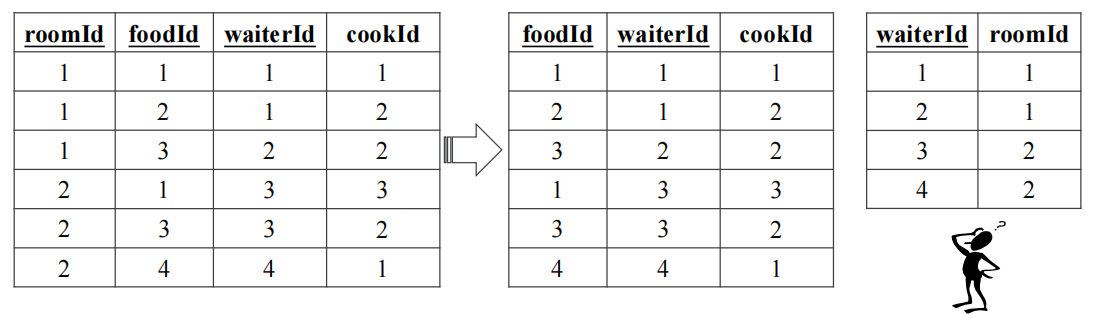
1. 3NF
   1. 要求：**关系模式中不能存在非主属性对码的传递依赖**

部分依赖是一种特殊的传递依赖（A,B→B→C）

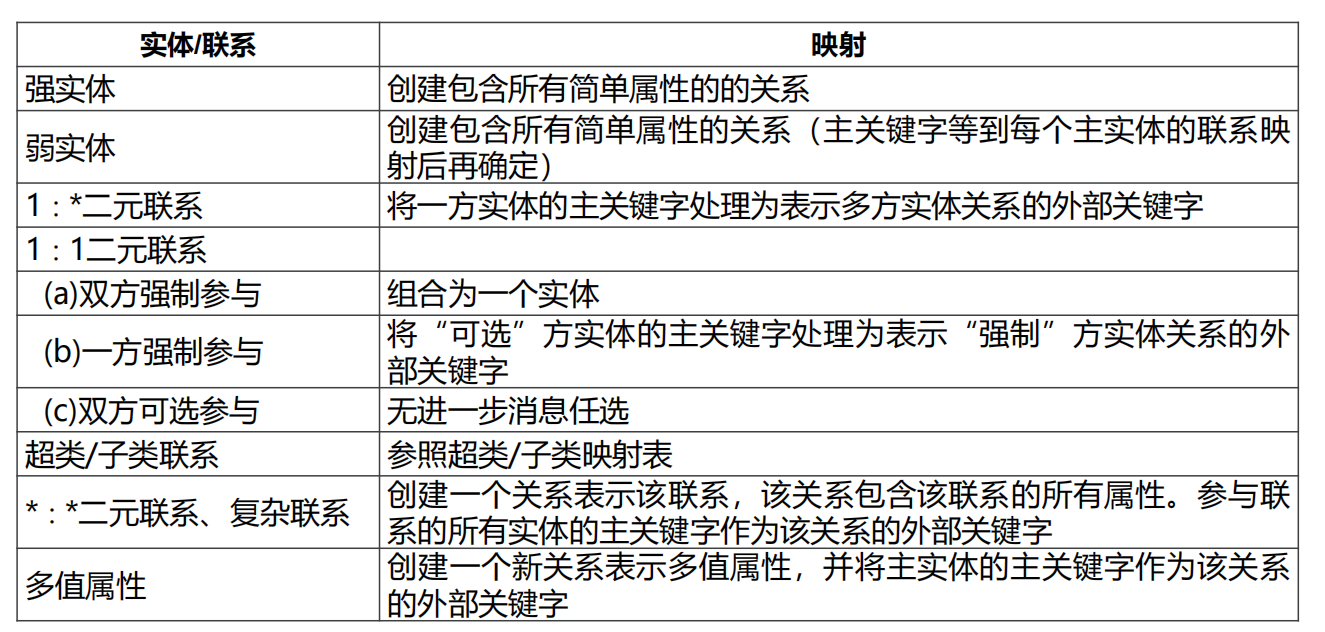
****

* 1. 问题：增删改异常问题
  2. 解决办法：分解——把传递依赖于码的属性连同它所依赖的属性分离出去，组成一个新的关系模式。同时，在原来的关系中要保留一个外键。

1. BCNF
   1. 要求：**关系模式中主属性也不能存在对码的部分依赖和传递依赖**
   2. 问题：冗余；增删改异常问题
   3. 解决办法：分解——把部分依赖或者传递依赖于码的主属性连同它所依赖的属性分离出去组成一个新的关系模式。



### ER模型向关系模型的映射（只放总结，具体实例见ppt）



## 事务管理

### 事务的概念和特性

1. 事务的概念

**事务是完成用户某一特定任务的与数据库的一次或多次交互的逻辑单位。**

事务（Transaction） 是对数据库进行访问或修改的一个或多个操作，这组操作组成一个单位，共同完成一个任务。（参考资料）

1. 事务ACID特性

原子性(Atomicity)：执行事务中的操作要么都做，要么都不做。

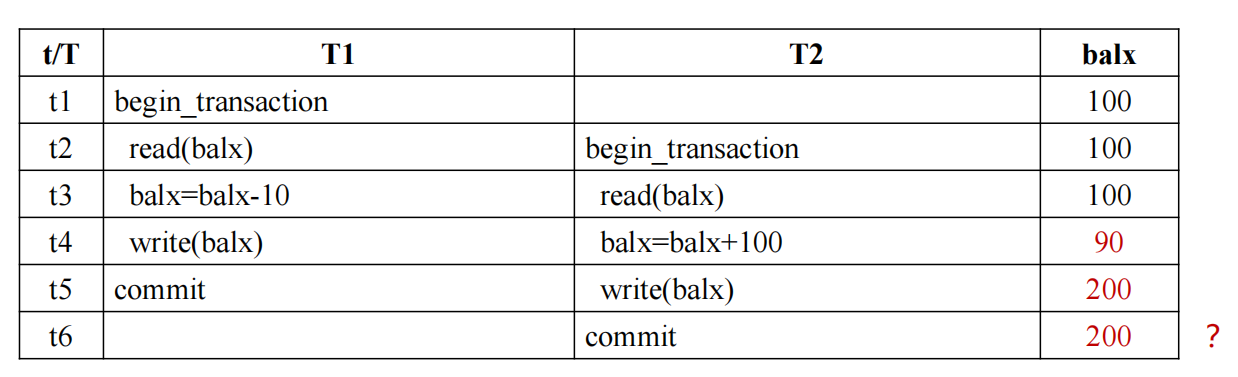
一致性(Consistency)：一致性要求事务维护数据库的完整性约束。

隔离性(Isolation)：并发执行的事务之间不能相互影响。

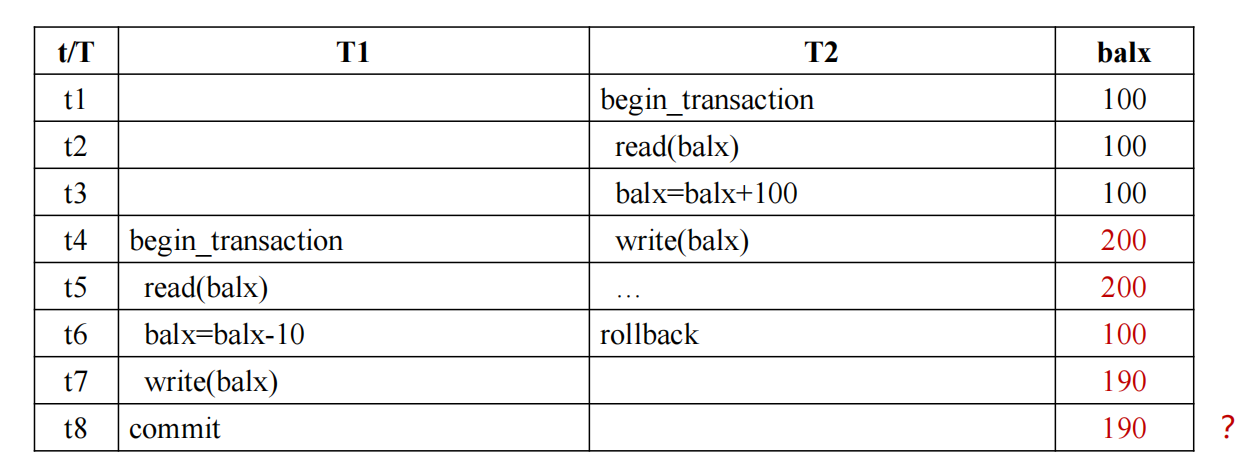
持续性(Durability)：事务一旦提交，它一定是永久生效的。

### 并发不加控制可能导致的问题

1. 丢失更新的问题：一个事务数据的更新会把另外一个事务对数据的更新给冲掉

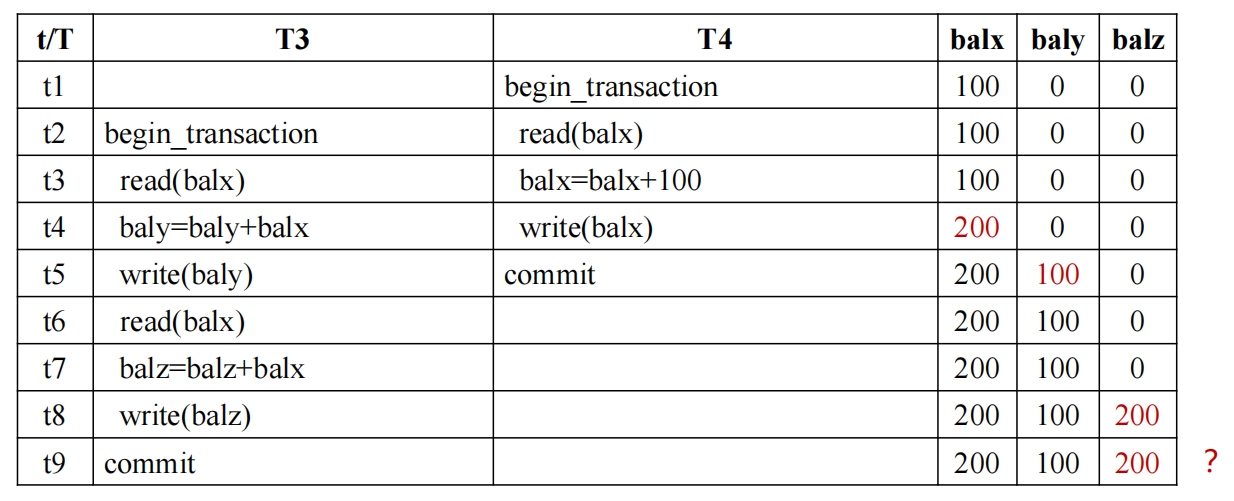


1. 脏读问题：一个事务读到了另外一个事务尚未提交的数据



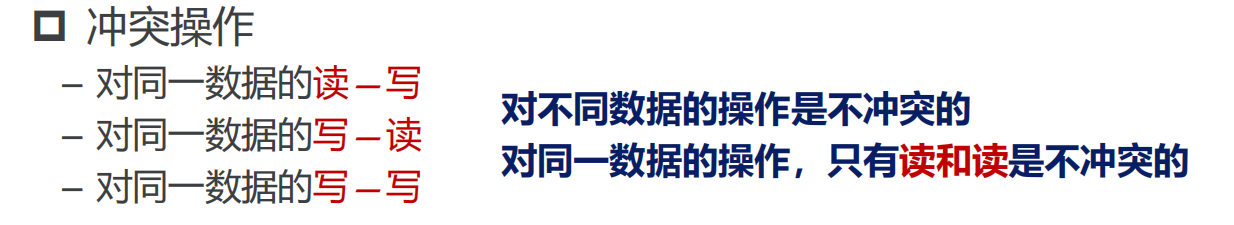
1. 不可重复读问题：多次读取同一数据，结果不一致。
2. 幻读问题：多次执行相同的查询，但结果集的行数发生了变化

（由于其他并发事务进行了数据插入、更新或删除操作，导致查询结果集的行数发生了幻影行的变化）



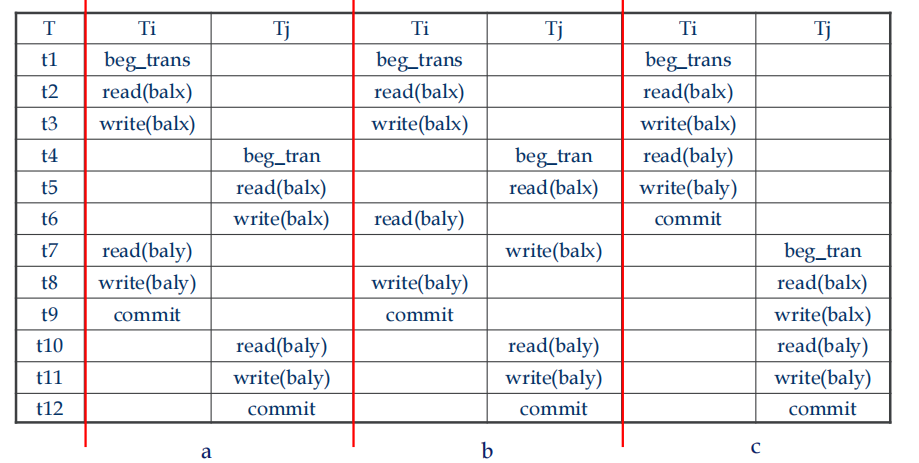
### 事务的调度及其基本概念

1. **调度：**一系列事务它们的操作相互交叉执行的序列称为调度。
2. **串行调度：**一种调度方式，其中每个事务的操作被连续执行，没有并发执行的情况。
3. **可串行化调度：**能够和某一个串行调度结果相同的调度称为可串行化调度。
4. **冲突可串行化调度：**若调度S可以通过交换某些相继非冲突操作变成某一串行调度，则称调度S为冲突可串行化调度。

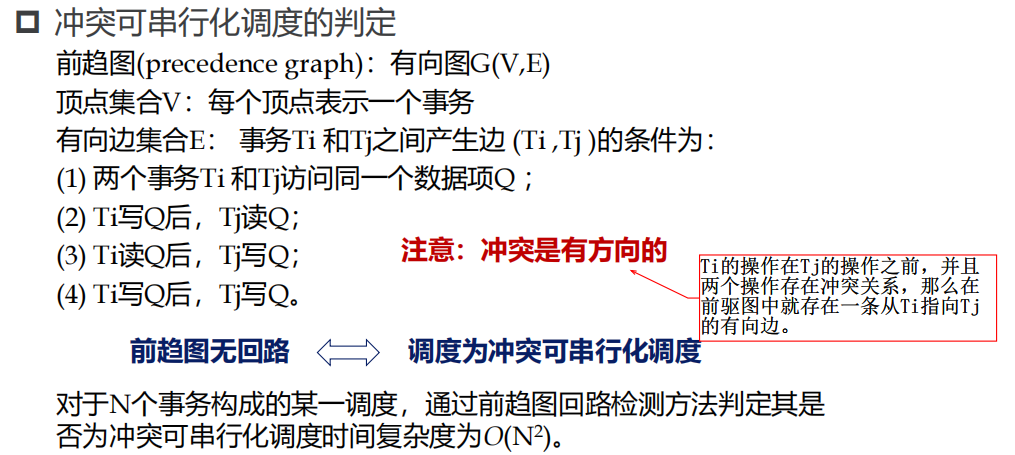


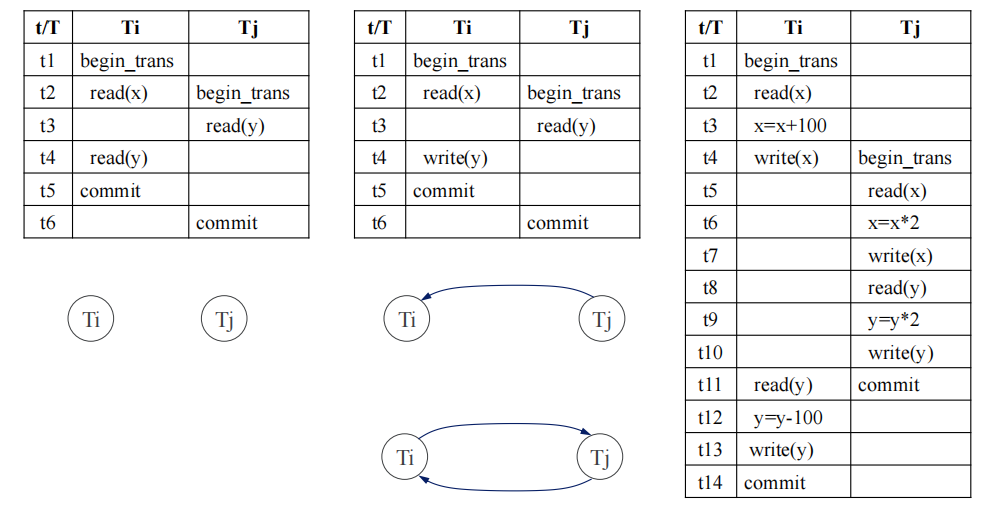


一个冲突可串行化调度的例子：



如何判断冲突串行化调度：



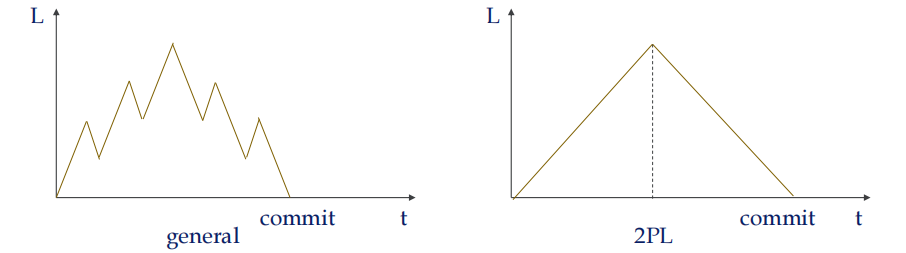


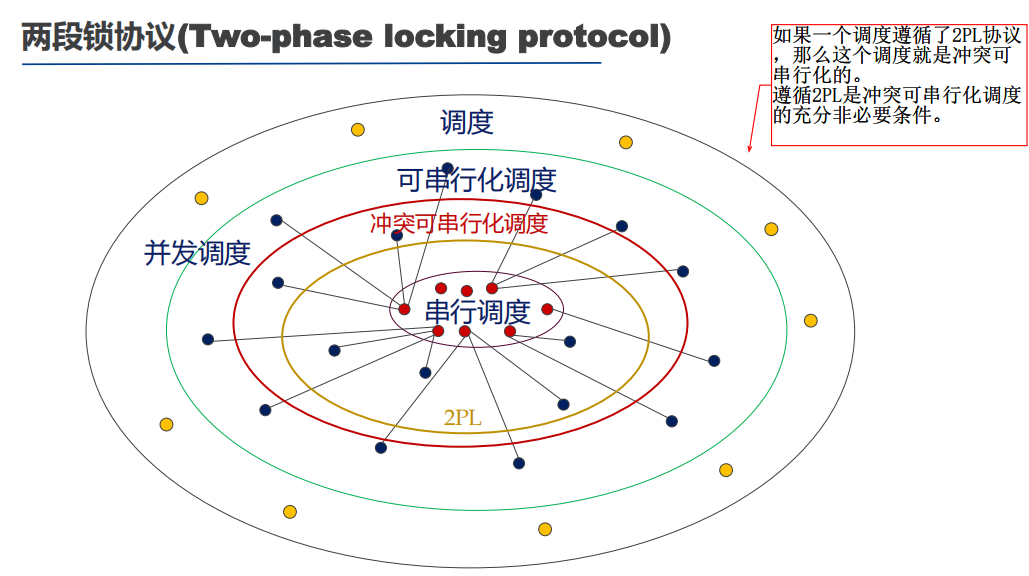
### 两段锁协议(Two-phase locking protocol)

1. 锁、共享锁、专用锁

* 锁(Lock)：控制事务并发访问数据库中数据的一种技术。
* 共享锁(Shared Lock)：也称为读锁，允许多个事务同时获取并持有锁，用于实现并发的读取操作。
* 专用锁(Exclusive lock)：也称为写锁，只允许一个事务独占地获取并持有锁，用于实现独占的写入或修改操作。

1. 2PL、Rigorous 2PL、Strict 2PL
   1. **2PL：每个事务的执行可以分为加锁阶段、解锁阶段两个阶段**





2PL存在的问题：回滚问题、下降点问题

* 1. **Rigorous 2PL：把锁的释放时机延迟，延迟到事务提交或者回滚前。**

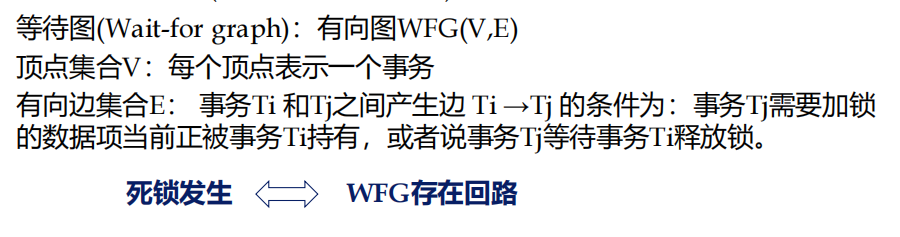
能够保证调度的冲突可串行化，即使最后事务不提交，回滚的话也能保证调度的冲突可串行化；但是降低了效率。

ps：回滚操作是指将已经执行的事务中的所有操作撤销并恢复到事务开始之前的状态的操作。当一个事务执行过程中发生错误或者被取消时，为了保持数据的一致性和完整性，需要对事务进行回滚。

* 1. **Strict 2PL：专用锁提交时再释放，共享锁可以提前释放。**

1. 死锁

* 两个事务相互等待对方释放锁的僵持局面叫做死锁
* 死锁的避免方法：资源的一次性分配、资源编号依次使用
* 死锁的检测：有向图
* 死锁的解除：中断机制、随机撤销一个事务



### 事务隔离等级含义，事务边界与隔离等级设计原则

事务隔离等级：多个并发事务之间相互隔离的程度，用于控制并发事务对数据的访问和修改的可见性。（PostgreSQL的隔离等级——提交读取、串行化）

* READ UNCOMMITTED

未提交读取（脏读）——有可能读到别的事务尚未提交的数据

* READ COMMIT

提交读取——只能读到别的事务已经提交的数据

* REPEATABLE READ

可重复读——在同一个事务中，对同一个数据的多次读，值是一样的（取决于第一次读取到的数据），即使在读的过程中别的事务是数据变化了，对你没有影响。

* SERIALIZABLE

串行化——对同一个数据是串行化执行的

1. 事务边界与隔离等级设计原则

事务的设计应该尽可能短小，尤其不应该把与用户、与设备交互的操作放到事务中。（业务面向用户，事务面向数据库）