第一章 绪论

数据库系统的构成

1. 数据库
2. 数据库管理系统（及其应用开发工具）
3. 应用程序
4. 数据库管理员

- 数据的整体结构化是数据库的主要特征之一
- 数据记录可以变长
    - 数据的最小存取单位是数据项
- 通俗地讲数据模型就是现实世界的模拟。
- 数据模型应满足三方面要求
    - 能比较真实地模拟现实世界
    - 容易为人所理解
    - 便于在计算机上实现
- 数据模型是数据库系统的核心和基础

- 数据模型分为两类（两个不同的层次）
    - 概念模型 也称信息模型
    - 逻辑模型和物理模型

1. 现实世界=>概念模型 数据库设计人员完成
2. 概念模型=>逻辑模型 数据库设计人员完成,数据库设计工具协助完成
3. 逻辑模型=>物理模型 由DBMS完成

- 实体内部的联系通常是指组成实体的各属性之间的联系
- 实体之间的联系通常是指不同实体集之间的联系
- 实体之间的联系有一对一、一对多和多对多等多种类型

1.2.3 数据模型的组成要素

- 数据结构
- 数据操作
- 数据的完整性约束条件

常用的数据模型

- 层次模型（Hierarchical Model）(树形结构)
- 网状模型（Network Model）(树形结构)
- 关系模型（Relational Model)）(二维表)

1.3.2 数据库系统的三级模式结构

- 模式（Schema）
- 外模式（External Schema）
- 内模式（Internal Schema）
- 模式（也称逻辑模式）
  - 数据库中全体数据的逻辑结构和特征的描述
  - 所有用户的公共数据视图
- 一个数据库只有一个模式
- 模式的地位：是数据库系统模式结构的中间层
- 模式的定义
  - 数据的逻辑结构（数据项的名字、类型、取值范围等）
  - 数据之间的联系
  - 数据有关的安全性、完整性要求
- 外模式（也称子模式或用户模式）
  - 数据库用户（包括应用程序员和最终用户）使用的**局 部**数据的逻辑结构和特征的描述
  - 数据库用户的数据视图，是与某一应用有关的数据的 逻辑表示
- 外模式的地位：介于模式与应用之间
  - 模式与外模式的关系：一对多
- 外模式通常是模式的子集
- 一个数据库可以有多个外模式。反映了不同的用户的应 用需求、看待数据的方式、对数据保密的要求
- 对模式中同一数据，在外模式中的结构、类型、长度、 保密级别等都可以不同
  - 外模式与应用的关系：一对多
- 同一外模式也可以为某一用户的多个应用系统所使用
- 但一个应用程序只能使用一个外模式
- 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
- 记录的存储方式（例如，顺序存储，按照B树结构存储， 按hash方法存储等）
- 索引的组织方式
- 数据是否压缩存储
- 数据是否加密
- 数据存储记录结构的规定
- 一个数据库只有一个内模式

数据库的二级映像功能与数据独立性

二级映象在数据库管理系统内部实现这三个抽象

- 层次的联系和转换
  - 外模式／模式映像
  - 模式／内模式映像

外模式／模式映像

- 模式：描述的是数据的全局逻辑结构
- 外模式：描述的是数据的局部逻辑结构
- 同一个模式可以有任意多个外模式
- 每一个外模式，数据库系统都有一个外模式／ 模式映象，定义外模式与模式之间的对应关系
- 映象定义通常包含在各自外模式的描述中保证数据的逻辑独立性
    - 当模式改变时，数据库管理员对外模式／模式映象作 相应改变，使外模式保持不变
    - 应用程序是依据数据的外模式编写的，应用程序不必 修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性

模式／内模式映像

- 模式／内模式映象定义了数据全局逻辑结构与 存储结构之间的对应关系。
    - 例如，说明逻辑记录和字段在内部是如何表示的
- 数据库中模式／内模式映象是唯一的
- 该映象定义通常包含在模式描述中
- 保证数据的物理独立性
    - 当数据库的存储结构改变了（例如选用了另一种存储 结构），数据库管理员修改模式／内模式映象，使模 式保持不变。
    - 应用程序不受影响。保证了数据与程序的物理独立性，简称数据的物理独立性
- 数据库模式
    - 即全局逻辑结构是数据库的中心与关键
    - 独立于数据库的其他层次
    - 设计数据库模式结构时应首先确定数据库的逻辑模式
- 数据库的内模式
    - 依赖于它的全局逻辑结构
    - 独立于数据库的用户视图，即外模式
    - 独立于具体的存储设备
    - 将全局逻辑结构中所定义的数据结构及其联系按照一 定的物理存储策略进行组织，以达到较好的时间与空 间效率

- 数据库的外模式
    - 面向具体的应用程序
    - 定义在逻辑模式之上
    - 独立于存储模式和存储设备
    - 当应用需求发生较大变化，相应外模式不能满足其视 图要求时，该外模式就得做相应改动
    - 设计外模式时应充分考虑到应用的扩充性
- 数据库的二级映像
    - 保证了数据库外模式的稳定性
    - 从底层保证了应用程序的稳定性，除非应用需求本身 发生变化，否则应用程序一般不需要修改
- 数据与程序之间的独立性，使得数据的定义和描 述可以从应用程序中分离出去

- 数据的存取由数据库管理系统管理
    - 简化了应用程序的编制
    - 大大减少了应用程序的维护和修改

1.5 小结

- 数据库系统概述
  - 数据库的基本概念
  - 数据管理的发展过程
  - 数据库系统的特点
- 数据模型
  - 数据模型的三要素
  - 三种主要数据库模型
- 数据库系统内部的系统结构
  - 数据库系统三级模式结构
  - 数据库系统两层映像系统结构

第二章 关系数据库

2.1 关系模型概述

关系

- 单一的数据结构----关系 现实世界的实体以及实体间的各种联系均用关系来表示
- 逻辑结构----二维表 从用户角度，关系模型中数据的逻辑结构是一张二维表

1. 域（Domain）
2. 笛卡尔积（Cartesian Product）
3. 关系(Relation)
4. 域是一组具有相同数据类型的值的集合
5. 分量（Component）
   - 笛卡尔积元素（d 1 ，d 2 ，…，d n ）中的每一个值d i 叫作一分量
6. 基数（Cardinal number） 最大的取值情况的数量。

7. 关系 D 1 ×D 2 ×…×D n 的子集叫作在域D 1 ，D 2 ，…，D n 上的 关系，表示为 R（D 1 ，D 2 ，…，D n ）

   - R： 关系名 ⊠n： 关系的目或度（Degree）(一元关系，二元关系，多元关系)

（5）属性

- 关系中不同列可以对应相同的域

（6）码

（7）三类关系 基本关系（基本表或基表） 实际存在的表，是实际存储数据的逻辑表示 查询表 查询结果对应的表 视图表 由基本表或其他视图表导出的表，是虚表，不对 应实际存储的数据

关系模式

- 关系模式（Relation Schema）是型
- 关系是值
- 关系模式是对关系的描述
  - 元组集合的结构 ☒ 属性构成 ☒ 属性来自的域 ☒ 属性与域之间的映象关系
  - 完整性约束条件

关系数据库

- 关系数据库
  - 在一个给定的应用领域中，所有关系的集合构成一 个关系数据库
- 关系数据库的型与值
  - 关系数据库的型: 关系数据库模式，是对关系数据 库的描述
  - 关系数据库的值: 关系模式在某一时刻对应的关系 的集合，通常称为关系数据库

2.2 关系操作

基本的关系操作

- 常用的关系操作
  - 查询操作：选择、投影、连接、除、并、差、交、笛卡 尔积 ☒ 选择、投影、并、差、笛卡尔积是5种基本操作
  - 数据更新：插入、删除、修改
- 关系操作的特点
  - 集合操作方式：操作的对象和结果都是集合，一次一集 合的方式

关系数据库语言的分类

- 关系代数语言
  - 用对关系的运算来表达查询要求
  - 代表：ISBL
- 关系演算语言：用谓词来表达查询要求
  - 元组关系演算语言 ☒ 谓词变元的基本对象是元组变量 ☒ 代表：APLHA, QUEL
  - 域关系演算语言 ☒ 谓词变元的基本对象是域变量 ☒ 代表：QBE
- 具有关系代数和关系演算双重特点的语言
  - 代表：SQL（Structured Query Language）

2.3 关系的完整性

关系模型必须满足的完整性约束条件称为关系的两个不变性。

- 规则2.1 实体完整性规则（Entity Integrity）
  - 若属性A是基本关系R的主属性，则属性A不能取空值

- 规则2.2 参照完整性规则 若属性（或属性组）F是基本关系R的外码它与基本关系S 的主码K s相对应（基本关系R和S不一定是不同的关系）， 则对于R中每个元组在F上的值必须为：
    - 或者取空值（F的每个属性值均为空值）
    - 或者等于S中某个元组的主码值
- 用户定义的完整性 针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求

例: 课程（课程号，课程名，学分）

- “课程号”属性必须取唯一值
- 非主属性“课程名”也不能取空值
- “学分”属性只能取值{1，2，3，4}

2.4 关系代数

- 传统的集合运算
    - 并 交 差 笛卡尔积
- 专门的关系运算
    - 选择 投影 连接 除

刷题：先看课件

1. **关系代数**
2. **元组关系演算语言ALPHA**

第三章 SQL

SQL的特点

1. 综合统一
2. 高度非过程化
3. 面向集合的操作方式
4. 以同一种语法结构提供多种使用方式
5. 语言简洁，易学易用

SQL功能极强，完成核心功能只用了9个动词。

SQL功能	动词
数据查询	SELECT
数据定义	CREATE，DROP，ALTER

SQL功能	动词
数据操纵	INSERT， UPDATE， DELETE
数据控制	GRANT，REVOKE

支持关系数据库三级模式结构

模式结构	对应SQL
外模式	视图
模式	基本表
内模式	存储文件

- 基本表
  - 本身独立存在的表
  - SQL中一个关系就对应一个基本表
  - 一个（或多个）基本表对应一个存储文件
  - 一个表可以带若干索引
- 存储文件
  - 逻辑结构组成了关系数据库的内模式
  - 物理结构对用户是隐蔽的
- 视图
  - 从一个或几个基本表导出的表
  - 数据库中只存放视图的定义而不存放视图对应的数据
  - 视图是一个虚表
  - 用户可以在视图上再定义视图

数据定义

SQL 的数据定义语句

操作对象	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEM	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

- 现代关系数据库管理系统提供了一个层次化的数据 库对象命名机制
  - 一个关系数据库管理系统的实例（Instance）中可以建立 多个数据库
  - 一个数据库中可以建立多个模式
  - 一个模式下通常包括多个表、视图和索引等数据库对象

数据查询

数据更新

空值的处理

视图

- 视图的特点
  - 虚表，是从一个或几个基本表（或视图）导出的表
  - 只存放视图的定义，不存放视图对应的数据
  - 基表中的数据发生变化，从视图中查询出的数据也

定义视图(建立与删除)

查询视图

更新视图

视图的作用

简角重构护晰表

- 视图能够简化用户的操作
- 视图使用户能以多种角度看待同一数据
- 视图对重构数据库提供了一定程度的逻辑独立性
- 视图能够对机密数据提供安全保护
- 适当的利用视图可以更清晰的表达查询

第四章 数据库安全性

- 数据库安全性控制的常用方法
  - 用户标识和鉴定
  - 存取控制
  - 视图
  - 审计
  - 数据加密

存取控制

1. 自主存取控制（用户对不同的数据对象以及不同用户对同一对象都有不同的权限，用户还可以转授权限）
2. 强制存取控制(每一数据对象被标以一定的密级，每个用户被授予某一级别的许可证)

自主存取控制

GRANT

```
grant <权限>
on <对象类型><对象名>
to <用户>
[with grant option]
```

all privileges 全部权限 public 全体成员 with grant option 允许再授予

e.g.

```
grant all privileges
on table Student,Course
to u2,u3
with grant option;
```

```
grant update(Sno), select
on table Student
to public;
```

REVOKE

```
revoke <权限>[,<权限>]...
on <对象类型> <对象名>[,<对象类型><对象名>]…
from <用户>[,<用户>]...[cascade | restrict];
```

级联收回权限

```
revoke insert
on table SC
from U5
cascade;
```

- 强制存取控制规则
- 仅当主体的许可证级别大于或等于客体的密级时，该主体才能读取相应的客体
- 仅当主体的许可证级别小于或等于客体的密级时，该主体才能写相应的客体

数据库角色

数据库角色：被命名的一组与数据库操作相关的 权限

1.角色的创建 CREATE ROLE <角色名> 2.给角色授权 GRANT <权限>[,<权限>]… ON <对象类型> 对象名 TO <角色>[,<角色>]…

3.将一个角色授予其他的角色或用户 GRANT <角色1>[,<角色2>]… TO <角色3>[,<用户1>]… [WITH ADMIN OPTION]

```
- 指定了WITH ADMIN OPTION则获得某种权限的角色或用
```

户还可以把这种权限授予其他角色

4.角色权限的收回 REVOKE <权限>[,<权限>]… ON <对象类型> <对象名> FROM <角色>[,<角色>]…

e.g.

1. 首先创建一个角色 R1

```
CREATE ROLE R1;
```

2. 然后使用GRANT语句，使角色R1拥有Student表的SELECT、UPDATE、INSERT权限

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO R1;
```

3. 将这个角色授予王平，张明，赵玲。使他们具有角色R1所包含的全部权限

```
GRANT R1
TO 王平,张明,赵玲;
```

4. 可以一次性通过R1来回收王平的这3个权限

```
REVOKE R1
FROM 王平;
```

视图机制

```
CREATE VIEW CS_Student
AS
SELECT *
FROM Student
WHERE Sdept='CS';
```

4.4 审计

- 什么是审计
  - 启用一个专用的审计日志（Audit Log） 将用户对数据库的所有操作记录在上面
  - 审计员利用审计日志 监控数据库中的各种行为，找出非法存取数据的人、时间和内容
- 加密方法
  - 存储加密 (内核级加密方法: 性能较好，安全完备性较高)
  - 传输加密

第五章 数据库完整性

3.违约处理

- 数据库管理系统若发现用户的操作违背了完整性约束条件，
- 数据库管理系统若发现用户的操作违背了完整性约束条件，

就采取一定的动作

- 拒绝（NO ACTION）执行该操作
- 级连（CASCADE）执行其他操作

完整性约束条件

primary key

1. 在列级定义主码

```
CREATE TABLE Student
( Sno CHAR(9) PRIMARY KEY,
```

2. 在表级定义主码

```
CREATE TABLE Student
( Sno CHAR(9),
Sname CHAR(20) NOT NULL,
Ssex CHAR(2),
Sage SMALLINT,
Sdept CHAR(20),
PRIMARY KEY (Sno)
);
```

参照完整性

foreign key (Sno) references Student(Sno),

```
CREATE TABLE SC
```

```
CREATE TABLE SC
( Sno CHAR(9) NOT NULL,

Cno CHAR(4) NOT NULL,
Grade SMALLINT,
PRIMARY KEY (Sno, Cno) , /*在表级定义实体完整性*/
FOREIGN KEY (Sno) REFERENCES Student(Sno),
/*在表级定义参照完整性*/
FOREIGN KEY (Cno) REFERENCES Course(Cno)
/*在表级定义参照完整性*/
);
```

5.3 用户定义的完整性

- TABLE时定义属性上的约束条件
  - 列值非空（NOT NULL）
  - 列值唯一（UNIQUE）
  - 检查列值是否满足一个条件表达式（CHECK）

```
Sname CHAR(8) NOT NULL,
Dname CHAR(9) UNIQUE NOT NULL,
/*要求Dname列值唯一，并且不能取空值*/
Ssex CHAR(2) CHECK （Ssex IN （‘男’,’女’））,
/*性别属性Ssex只允许取'男'或'女' */
Grade SMALLINT CHECK (Grade>=0 AND Grade <=100),
```

第六章 关系数据理论

6.2 规范化

6.2.1 函数依赖

1.函数依赖 2.平凡函数依赖与非平凡函数依赖 3.完全函数依赖与部分函数依赖

- **f和p**

4.传递函数依赖

- 箭头上有“传递”

6.2.2 码

候选码

- 找候选码的方法：
- 只出现在左边的一定在所有候选码之中，只出现在右边的不存在于所有候选码。

6.2.3 范式

6.2.4 2NF

- 定义6.6 若关系模式R∈1NF，并且每一个非主属性

6.2.5 3NF

- 定义6.7 设关系模式R∈1NF,若R中不存在 这样的码X、属性组Y及非主属性Z（Z ⊈Y）, 使 得X→Y，Y→Z成立，Y ↛X不成立，则称 R ∈ 3NF。
  - SC没有传递依赖，因此SC ∈ 3NF
  - S-L中Sno →Sdept(Sdept ↛ Sno), Sdept→Sloc，<>

6.2.6 BCNF

- BCNF（Boyce Codd Normal Form）由Boyce 和Codd提出，比3NF更进了一步。通常认为BCNF是修正的第三范式，有时也称为扩充的第 三范式。
- 定义6.8 设关系模式R∈1NF，若X →Y且Y ⊆ X时X必含有码，则R∈BCNF。
- 换言之，在关系模式R中，如果每一个决定
- 换言之，在关系模式R中，如果每一个决定 属性集都包含候选码，则R∈BCNF。<>

6.2.7 多值依赖

- 按照语义对于W的每一个值W i ，S有一个完整的集 合与之对应而不问C取何值。所以W→→S。

6.2.8 4NF

- 定义6.10 关系模式R∈1NF，如果对于R 的每个非平凡多值依赖X→→Y（Y ⊈ X），X都 含有码，则R∈4NF。
- 4NF就是限制关系模式的属性之间不允许有非平 凡且非函数依赖的多值依赖。4NF所允许的非平 凡多值依赖实际上是函数依赖。<>

6.2.9 规范化小结

- 一个低一级范式的关系模式，通过模式分解可以 转换为若干个高一级范式的关系模式集合，这种 过程就叫关系模式的规范化。
- 关系数据库的规范化理论是数据库逻辑设计的工

第七章 数据库设计

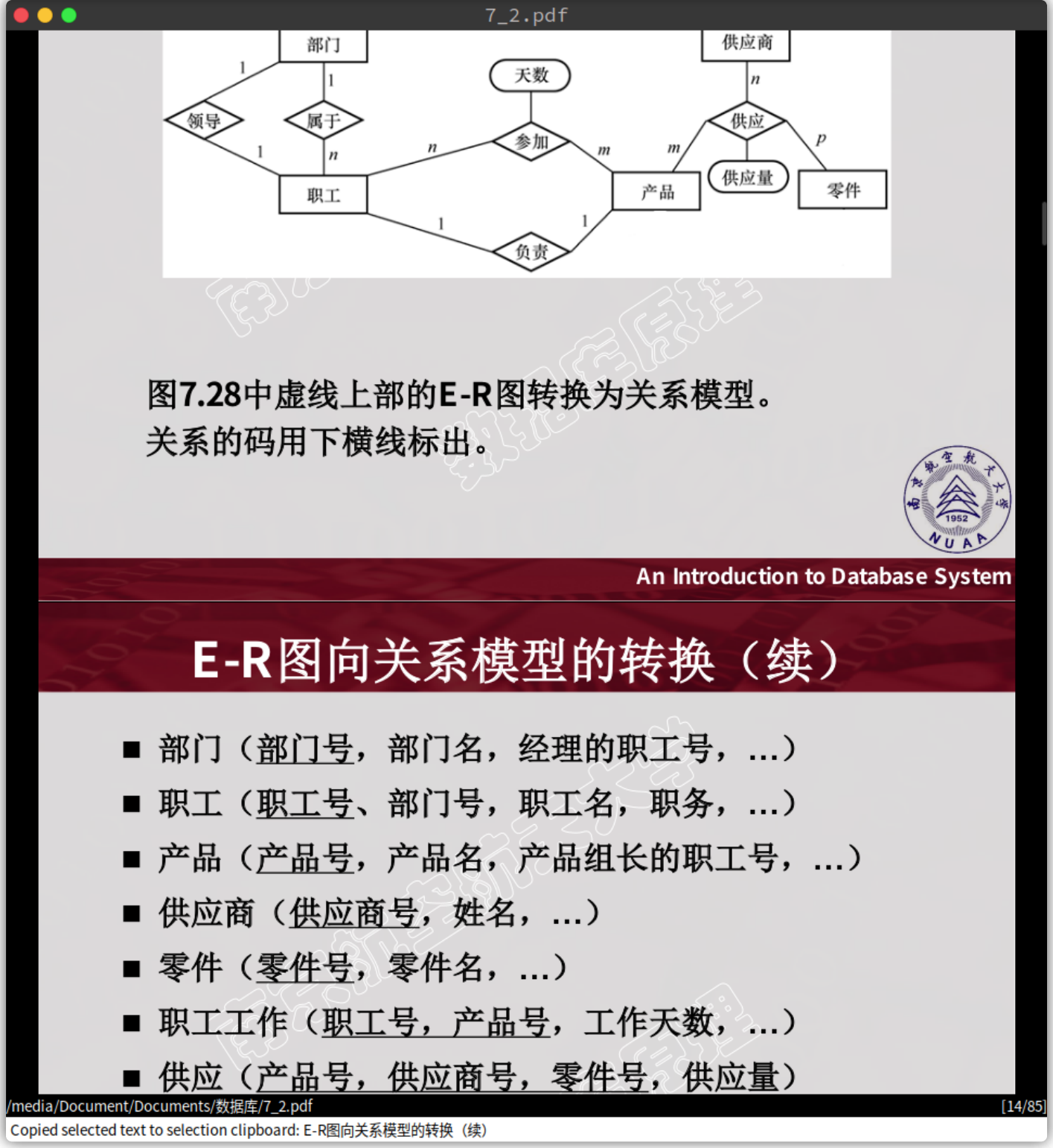
- 数据库设计分6个阶段
  - 需求分析
  - 概念结构设计(E-R图)
  - 逻辑结构设计(数据模型)
  - 物理结构设计(存取方法选择和存取路径建立)
  - 数据库实施
  - 数据库运行和维护

- 数据字典的内容
  - 数据项
  - 数据结构
  - 数据流
  - 数据存储
  - 处理过程
- 数据项是数据的最小组成单位
- 若干个数据项可以组成一个数据结构
- 数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容
- 对数据项的描述

数据项描述={数据项名,数据项含义说明,别名, 数据类型,长度,取值范围,取值含义, 与其他数据项的逻辑关系, 数据项之间的联系}

E-R图

E-R图想关系模型的转换



第八章 数据库编程

- 为什么要引入嵌入式SQL
  - SQL语言是非过程性语言
  - 事务处理应用需要高级语言

- 主语言
  - 嵌入式SQL是将SQL语句嵌入程序设计语言中，被嵌入 的程序设计语言，如C、C++、Java，称为主语言
- 在SQL语句中使用的主语言程序变量简称为主变量
- 游标是系统为用户开设的一个数据缓冲区，存放SQL
- 存储过程的优点

- 运行效率高
- 降低了客户机和服务器之间的通信量
- 方便实施企业规则
- 函数和存储过程的异同
  - 同：都是持久性存储模块
  - 异：函数必须指定返回的类型
- ODBC优点
  - 移植性好
  - 能同时访问不同的数据库

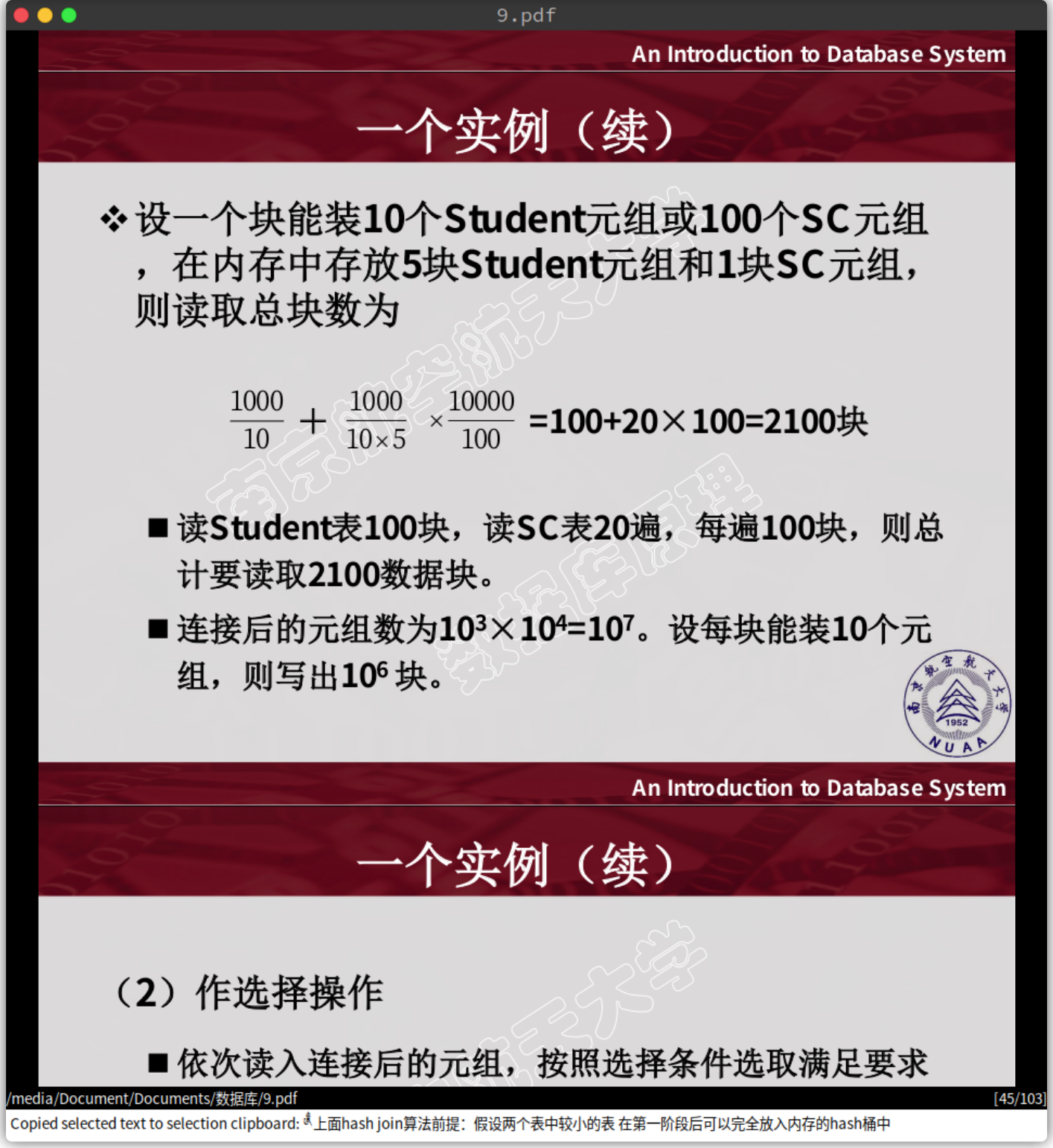
第九章 关系查询处理和查询优化

- 本章内容：
  - 关系数据库管理系统的查询处理步骤
  - 查询优化的概念
  - 基本方法和技术
- 查询优化分类 ：
  - 代数优化：指关系代数表达式的优化
  - 物理优化：指存取路径和底层操作算法的选择

9.1 关系数据库系统的查询处理

- 选择操作典型实现方法：
  i. 全表扫描方法 (Table Scan)
  ii. 索引扫描方法 (Index Scan)
- 连接操作的实现
  i. 嵌套循环算法(nested loop join)
  ii. 排序-合并算法(sort-merge join 或merge join)
  iii. 索引连接(index join)算法
  iv. Hash Join算法
  v. 上面hash join算法前提：假设两个表中较小的表在第一阶段后可以完全放入内存的hash桶中

9.2 关系数据库系统的查询优化



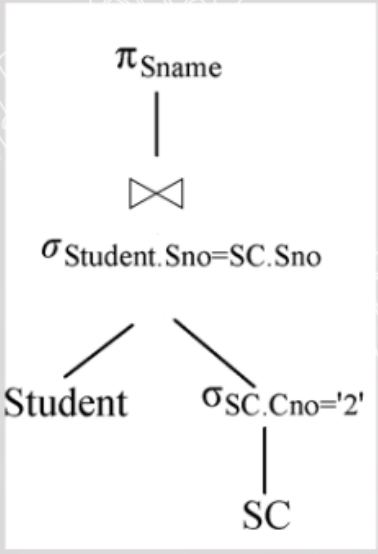
9.3 代数优化

典型的启发式规则

1. 选择运算应尽可能先做。在优化策略中这是最重要、最基本的一条。
2. 把投影运算和选择运算同时进行如有若干投影和选择运算，并且它们都对同一个关系操作，则可以在扫描此关系的同时完成所有的这些运算以避免重复扫描关系。
3. 把投影同其前或其后的双目运算结合起来，没有必要为了去掉某些字段而扫描一遍关系。
4. 把某些选择同在它前面要执行的笛卡尔积结合起来成为一个连接运算，连接特别是等值连接运算要比同样关系上的笛卡尔积省很多时间。
5. 找出公共子表达式

（2）对查询树进行优化

■ 利用规则4、6把选择$\sigma_{SC.Cno='2'}$移到叶端，图9.4查询树便转换成下图优化的查询树。这就是9.2.2节中Q3的查询树表示。



9.4 物理优化

代价估算

1. 全表扫描算法的代价估算公式
   - 如果基本表大小为B块，全表扫描算法的代价 cost＝B
   - 如果选择条件是“码＝值”，那么平均搜索代价 cost＝B/2
2. 索引扫描算法的代价估算公式
   - 如果选择条件是“码＝值”
   - 则采用该表的主索引
   - 若为B+树，层数为L，需要存取B+树中从根结点到叶 结点L块，再加上基本表中该元组所在的那一块，所 以cost=L+1
   - 如果选择条件涉及非码属性
   - 若为B+树索引，选择条件是相等比较，S是索引的选 择基数(有S个元组满足条件)
   - 满足条件的元组可能会保存在不同的块上，所以(最坏 的情况)cost=L+S
   - 如果比较条件是＞，＞＝，＜，＜＝操作
   - 假设有一半的元组满足条件
   - 就要存取一半的叶结点
   - 通过索引访问一半的表存储块 cost=L+Y/2+B/2
   - 如果可以获得更准确的选择基数，可以进一步修正 Y/2与B/2
3. 嵌套循环连接算法的代价估算公式
   - 嵌套循环连接算法的代价 cost＝Br+BrBs/(K-1)
   - 如果需要把连接结果写回磁盘 cost＝Br+Br Bs/(K-1)+(Frs*Nr*Ns)/Mrs
   - 其中Frs为连接选择性(join selectivity)，表示连接结果元组数的比例,Mrs是存放连接结果的块因子，表示每块中可以存放的结果元组数目

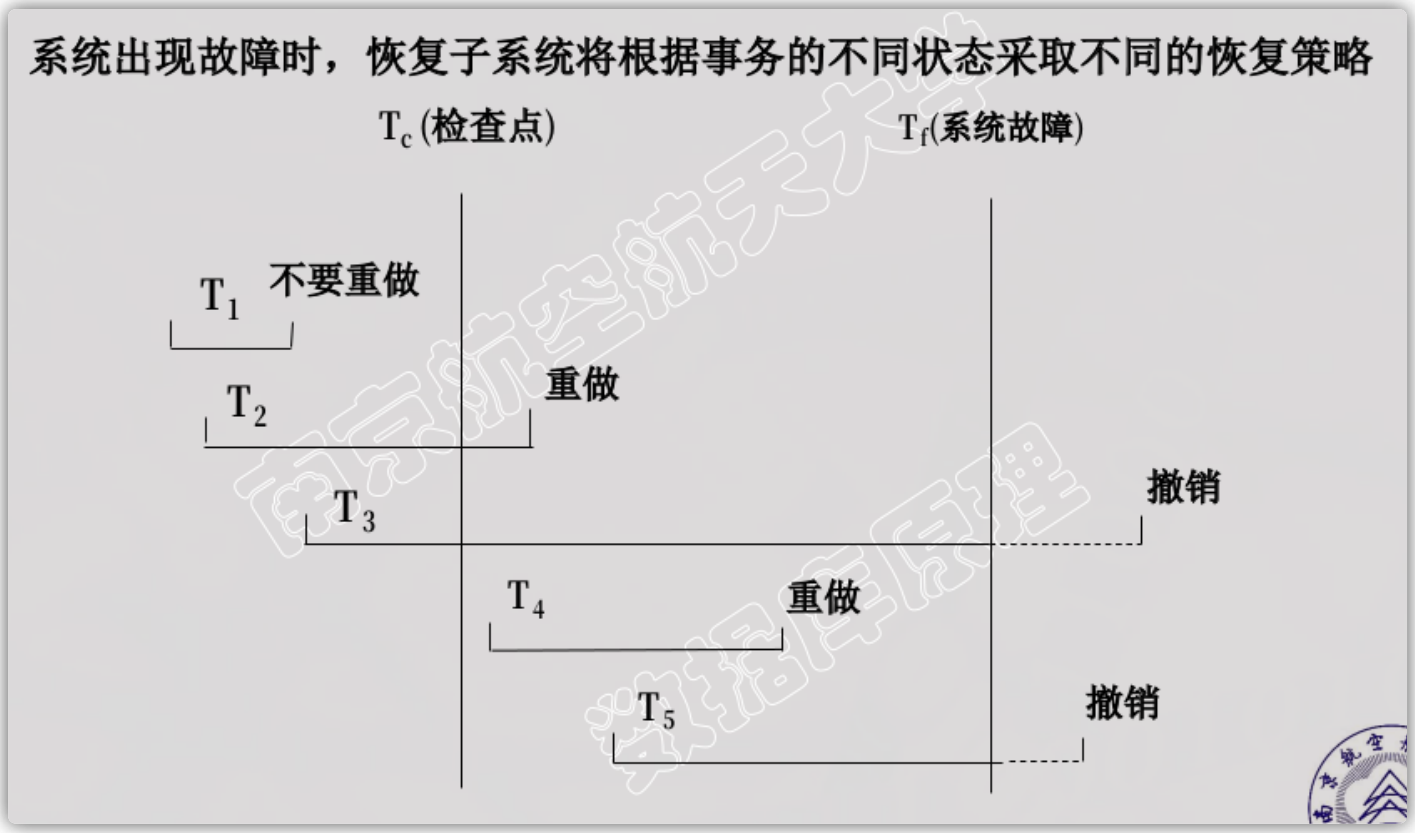
4. 排序-合并连接算法的代价估算公式
    - 如果连接表已经按照连接属性排好序，则 cost＝Br+Bs+(Frs*Nr*Ns)/Mrs
    - 如果必须对文件排序
    - 还需要在代价函数中加上排序的代价
    - 对于包含B个块的文件排序的代价大约是

10 事务

- 事务(Transaction)是用户定义的一个数据库操作序列， 这些操作要么全做，要么全不做，是一个不可分割的工作 单位。
- 事务是恢复和并发控制的基本单位

事务的ACID特性：

- 原子性（Atomicity）
    - 事务是数据库的逻辑工作单位
    - 事务中包括的诸操作要么都做，要么都不做
- 一致性（Consistency）
    - 事务执行的结果必须是使数据库从一个一致性状态变 到另一个一致性状态
- 隔离性（Isolation）
- 持续性（Durability ）
    - 一个事务的执行不能被其他事务干扰
- 持续性也称永久性（Permanence）
    - 一个事务一旦提交，它对数据库中数据的改变就应该 是永久性的。
    - 接下来的其他操作或故障不应该对其执行结果有任何



利用检查点的恢复步骤

1. 从重新开始文件中找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录
2. 由该检查点记录得到检查点建立时刻所有正在执行的事务清单ACTIVE-LIST
    - 建立两个事务队列 - UNDO-LIST - REDO-LIST
    - 把ACTIVE-LIST暂时放入UNDO-LIST队列，REDO队列暂为空。
3. 从检查点开始正向扫描日志文件，直到日志文件结 束
    - 如有新开始的事务T i ，把T i 暂时放入UNDO-LIST队列
    - 如有提交的事务T j ，把T j 从UNDO-LIST队列移到REDO-LIST队列;直到日志文件结束
4. 对UNDO-LIST中的每个事务执行UNDO操作,对REDO-LIST中的每个事务执行REDO操作

11.5 并发调度的可串行性

- 数据库管理系统对并发事务不同的调度可能会产 生不同的结果
- 串行调度是正确的
- 执行结果等价于串行调度的调度也是正确的，称 为可串行化调度

- 可串行化(Serializable)调度
    - 多个事务的并发执行是正确的，当且仅当其结果与
- 冲突操作：是指不同的事务对同一数据的读写操 作和写写操作：
    - R i (x)与W j (x)
    - W i (x)与W j (x)
- 不能交换（Swap）的动作:
    - 同一事务的两个操作
    - 不同事务的冲突操作
- 一个调度Sc在保证冲突操作的次序不变的情况下， 通过交换两个事务不冲突操作的次序得到另一个调 度Sc’，如果Sc’是串行的，称调度Sc是冲突可串行 化的调度

- 若一个调度是冲突可串行化，则一定是可串行化的 调度

- 冲突可串行化调度是可串行化调度的充分条件，不是必要条件。还有不满足冲突可串行化条件的可串行化调度。