数据库技术

作成者:周东

作成日:2019.08.29

Neusoft

目录

Neusoft

数据库技术基础

基础概念：

数据(Data)
数据库(DataBase)
数据库管理系统(DBMS)
数据库系统(DBS)

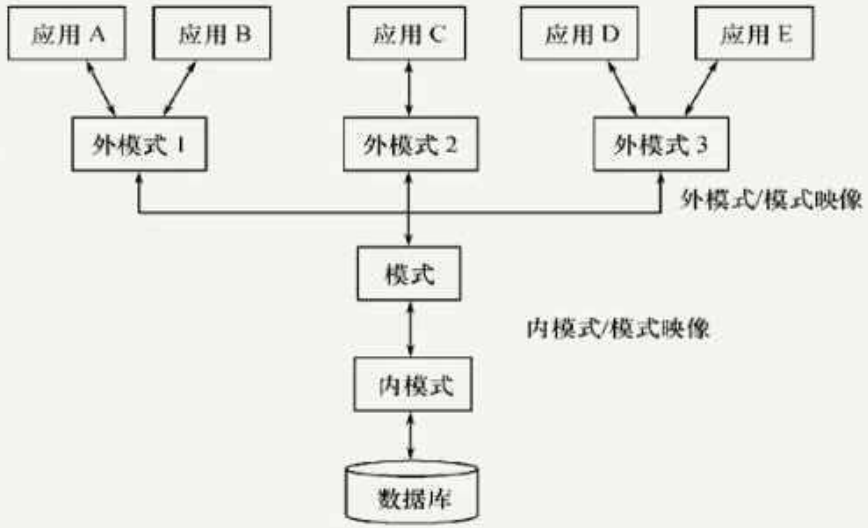
数据库技术基础

数据库三级模式结构

数据库系统的设计目标是允许用户逻辑地处理数据，而不必涉及这些数据在计算机中是怎样存放的，在数据组织和用户应用之间提供某种程度的独立性。

数据库系统可以分为：

- 外模式(子模式、用户模式)
- 模式(概念模式、逻辑模式)
- 内模式(存储模式)



数据模型

基本概念

①概念数据模型

E-R模型

②基本数据模型

层次模型
网状模型
关系模型
面向对象模型

数据模型

数据模型三要素

- 数据结构
- 数据操作
- 数据的约束条件

数据模型

E-R模型

- 实体
- 属性
- 实体集
- 域
- 实体型
- 码
- 联系

数据模型

E-R模型

实体:客观存在并可相互区别的事物称。可以是具体的人、事、物或抽象的概念。
客观实体:人、汽车、图书
抽象实体:账户、贷款

属性:实体所具有的某一特性称为属性。一个实体可以由若干个属性来刻画。
属性分类:
简单属性和复合属性
单值属性和多值属性
NULL属性
派生属性

实体集:具有相同类型和共享相同属性的实体的集合。

域:属性的取值范围称为该属性的域。

实体型:用实体名及其属性名集合来抽象和刻画, 同类实体称为实体型。

码:唯一标识实体的属性集称为码。

数据模型

E-R模型

联系:现实世界中事物内部以及事物之间的联系，在信息世界中反映为实体内部的联系和实体之间的联系。
两个实体型间联系分为三类
- 一对一联系
- 一对多联系
- 多对多联系

关系数据库

概念

在用户观点下，关系模型中数据的逻辑结构是一张二维表，它由行和列组成。

优点：
建立在严格的数学概念基础上。
概念单一，结构简单、清晰，用户易懂易用。
存取路径对用户透明，从而数据独立性、安全性好，简化数据库开发工作。

缺点：
由于存取路径透明，查询效率往往不如非关系数据模型。

关系数据库

关系的三种类型

①基本关系
②查询表
③视图

关系的约束

①实体完整性
②参照完整性
③用户定义完整性

关系数据库

查询优化准则:

①提早执行选择运算。目的:减少中间结果。
②合并乘积与选择运算为连接运算。目的:避免扫描大的关系。
③将投影运算与其它运算同时进行。目的:避免重复扫描关系。
④将投影运算与二目运算结合起来。目的:减少扫描关系的遍数。
⑤在执行连接前对关系适当地预处理:索引连接法, 排序合并连接法。
⑥存储公共子表达式。目的:只需检索中间结果, 不需重复计算。

关系数据库

函数依赖

部分函数依赖
完全函数依赖
传递函数依赖

关系数据库

规范化理论

第一范式(1NF)
如果关系模式R的每个关系r的属性值都是不可分的原子值，那么称R是第一范式的模式，r是规范化的关系。

第二范式(2NF)
若关系模式R是1NF，且每个非主属性完全函数依赖于候选键，那么称R是2NF模式。

第三范式(3NF)
如果关系模式R是1NF，且每个非主属性都不传递依赖于R的候选码，则称R是3NF模式。

BC范式(BCNF)
若关系模式R是1NF，且每个属性都不传递依赖于R的候选键，那么称R是BCNF模式。

SQL语言

数据库语言概述

数据定义语言DDL
负责数据结构定义与数据库对象定义的语言，由CREATE、ALTER与DROP三个语法组成。

数据操纵语言DML
实现对数据库的基本操作。如，对表中数据的SELECT、INSERT、DELETE和UPDATE。

SQL语言

创建表(CREATE TABLE)

语句格式:

CREATE TABLE<表名> (<列名> <数据类型> [列级完整性约束条件] [, <列名> <数据类型> [列级完整性约束条件]] ..... [, <表级完整性约束条件>]);

```
CREATE TABLE Student (
id     varchar(10),
name   varchar(20),
PRIMARY KEY (id)
);
```

SQL语言

修改表(ALTER TABLE)

语句格式:

ALTER TABLE <表名> [ADD <新列名> <数据类型> [完整性约束条件]] [DROP <完整性约束名>] [MODIFY <列名> <数据类型>];

ALTER TABLE Student ADD MailAddress varchar(30);

SQL语言

删除表(DROP TABLE)

语句格式:

DROP TABLE <表名>;

DROP TABLE Student;

SQL语言

创建索引(CREATE INDEX)

语句格式:

CREATE [UNIQUE] [CLUSTER] INDEX <索引名> ON <表名> (<列名> [<次序>] [, <列名> [<次序>]]……);

在创建索引的时候，遵循以下的经验性原则：

- 在经常需要搜索的列上建立索引
- 在主关键字上建立索引
- 在经常用于连接的列上建立索引
- 在经常需要根据范围进行搜索的列上建立索引
- 在经常需要排序的列上建立索引
- 在经常成为查询条件的列上建立索引

SQL语言

创建视图(CREATE VIEW)

语句格式:

CREATE VIEW <视图名> (列表名) AS SELECT 查询子句 [WITH CHECK OPTION];

SQL语言

数据库的查询(SELECT)

语句格式:

SELECT [ALL | DISTINCT] <目标列表达式> [, <目标列表达式>] ……
  FROM <表名或视图名> [, <表名或视图名>]
  [WHERE <条件表达式>]
  [GROUP BY <列名1> [HAVING <条件表达式>]]
  [ORDER BY <列名2> [ASC | DESC] ……];

运算符		含义	运算符		含义
集合成员运算符	IN NOT IN	在集合中 不在集合中	算术运算符	> ≥ < ≤ = ≠	大于 大于等于 小于 小于等于 等于 不等于
字符串匹配运算符	LIKE	与_和%进行单个多个字符匹配			
空值比较运算符	IS NULL IS NOT NULL	为空 不能为空	逻辑运算符	AND OR NOT	与 或 非

SQL语言

数据库的查询(SELECT)

连接查询:

```
SELECT s.id, s.name, c.name
  FROM Student s, Class c
WHERE s.id = c.s_id;
```

SQL语言

数据库的查询(SELECT)

子查询(嵌套查询):

```
SELECT s.id, s.name
  FROM Student s, Class c
WHERE s.id IN (SELECT s_id FROM sc WHERE score >= 60);
```

SQL语言

聚集函数

平均值:AVG
最小值:MIN
最大值:MAX
求和 :SUM
计数 :COUNT

SQL语言

数据库的查询(SELECT)

分组查询(GROUP BY):

```
SELECT s_id, AVG(score)
    FROM sc
GROUP BY s_id;
```

HAVING子句:

```
SELECT s_id, AVG(score)
    FROM sc
GROUP BY s_id HAVING AVG(score >= 60);
```

SQL语言

其它语法总结：

更名操作(AS)：

```
SELECT id AS “学号”,
            name AS “姓名”
    FROM Student;
```

字符串匹配(LIKE)：

```
SELECT id AS “学号”,
            name AS “姓名”
    FROM Student
 WHERE id like ‘%2019%’;
```

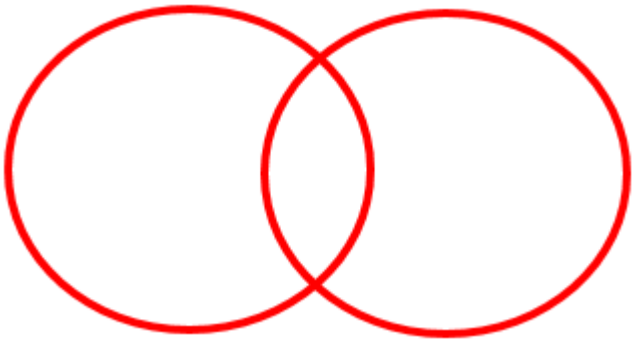
```
SELECT id AS “学号”,
            name AS “姓名”
    FROM Student
 WHERE name like ‘李_’;
```

SQL语言

其它语法总结：

集合操作：

并：UNION
交：INTERSECT
差：EXCEPT



授权与销权

授权语句

GRANT <权限> [, <权限>].....[ON <对象类型> <对象名>] TO <用户> [, <用户>].....[WITH GRANT OPTION];

对象	对象类型	操作权限
属性列	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES(5 种权限的总和)
视图	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES(5 种权限的总和)
基本表	TABLE	SELECT，INSERT，UPDATE，DELETE，ALTER，INDEX ALL PRIVILEGES(6 种权限的总和)
数据库	DATABASE	CREATETAB 建立表的权限，可由 DBA 授予普通用户

授权与销权

销权语句

REVOKE <权限> [, <权限>].....[ON <对象类型> <对象名>] FROM <用户> [, <用户>].....;

触发器(Trigger)

触发器的触发事件

种　类	关 键 字	含　义
DML 事件(3 种)	INSERT	在表或视图中插入数据时触发
	UPDATE	修改表或视图中的数据时触发
	DELETE	在删除表或视图中的数据时触发
DDL 事件(3 种)	CREATE	在创建新对象时触发
	ALTER	修改数据库或数据库对象时触发
	DROP	删除对象时触发
数据库事件(5 种)	STARTUP	数据打开时触发
	SHUTDOWN	在使用 NORMAL 或 IMMEDIATE 选项关闭数据库时触
	LOGON	当用户连接到数据库并建立会话时触发
	LOGOFF	当一个会话从数据库中断开时触发
	SERVERERROR	发生服务器错误时触发

触发器的触发时间

- Before：事件发生之前触发
- After：事件发生之后触发

触发器(Trigger)

创建触发器

```
CREATE TRIGGER <触发器名> [{BEFORE | AFTER}]
   {[DELETE | INSERT | UPDATEOF [列名清单]]}
   ON 表名
   [REFERENCING <临时视图名>]
   [WHEN <触发条件>]
BEGIN
   <触发动作>
END [触发器名];
```

触发器(Trigger)

更改触发器

```
ALTER TRIGGER <触发器名> [{BEFORE | AFTER}]
   {[DELETE | INSERT | UPDATEOF [列名清单]]}
   ON 表名
   AS
BEGIN
   <触发动作>
END [触发器名];
```

触发器(Trigger)

删除触发器

DROP TRIGGER <触发器名>;

事务

事务是用户定义的一个数据操作序列，这些操作序列要么全做要么全都不做，是一个不可分割的工作单位。

事务具有以下特性：
①原子性
②一致性
③隔离性
④持续性

DBMS运行的基本工作单位是事务。

事务

事务开始：

BEGIN TRANSACTION

事务提交：

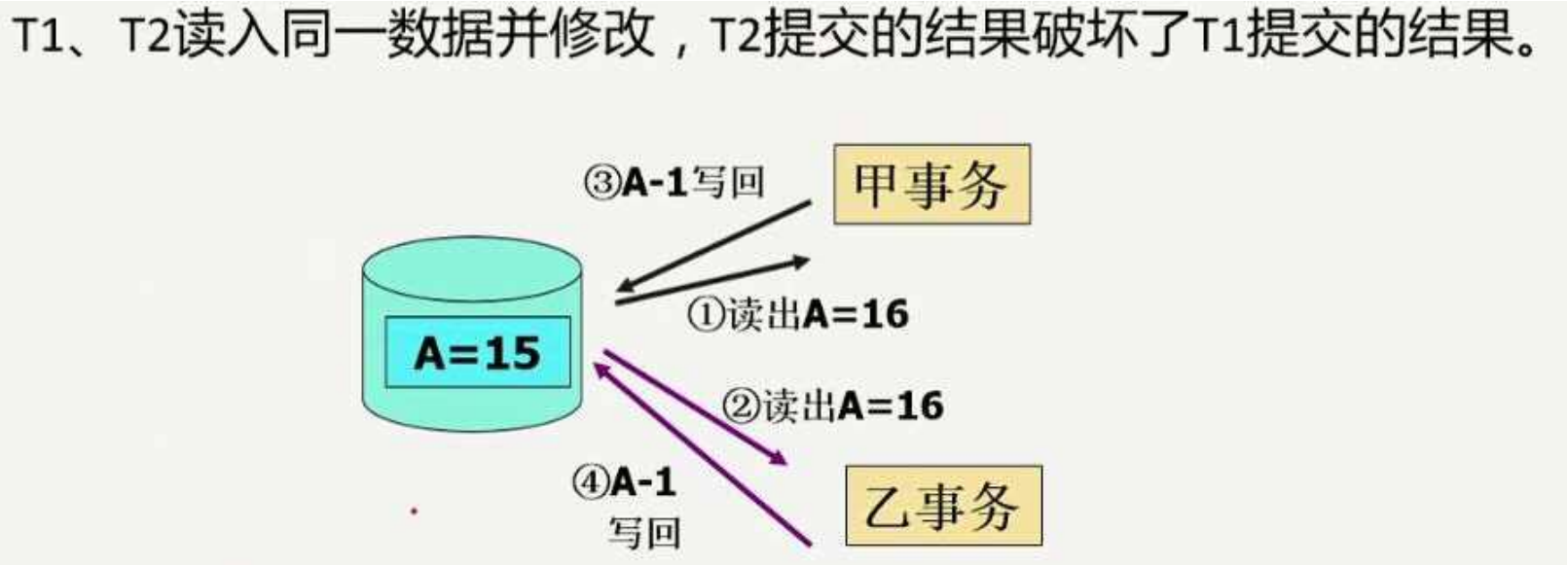
COMMIT

事务回滚：

ROLLBACK

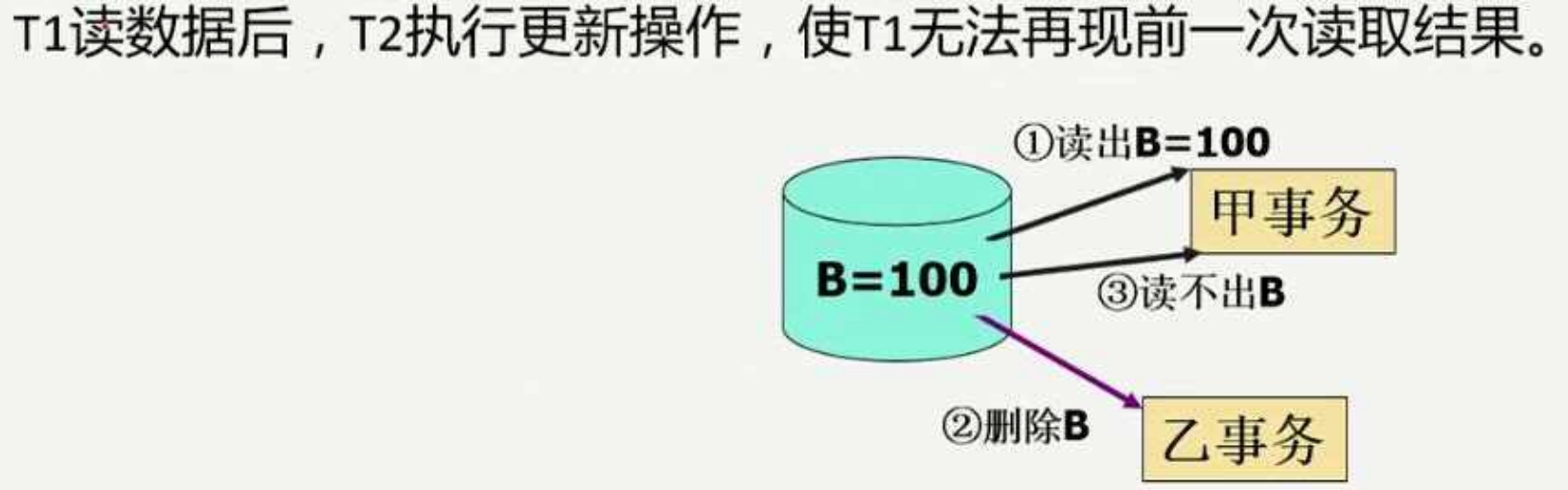
事务

丢失更新:


T1、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。
③A-1写回
甲事务
①读出A=16
A=15
②读出A=16
④A-1
写回
乙事务

事务

不可重复读:

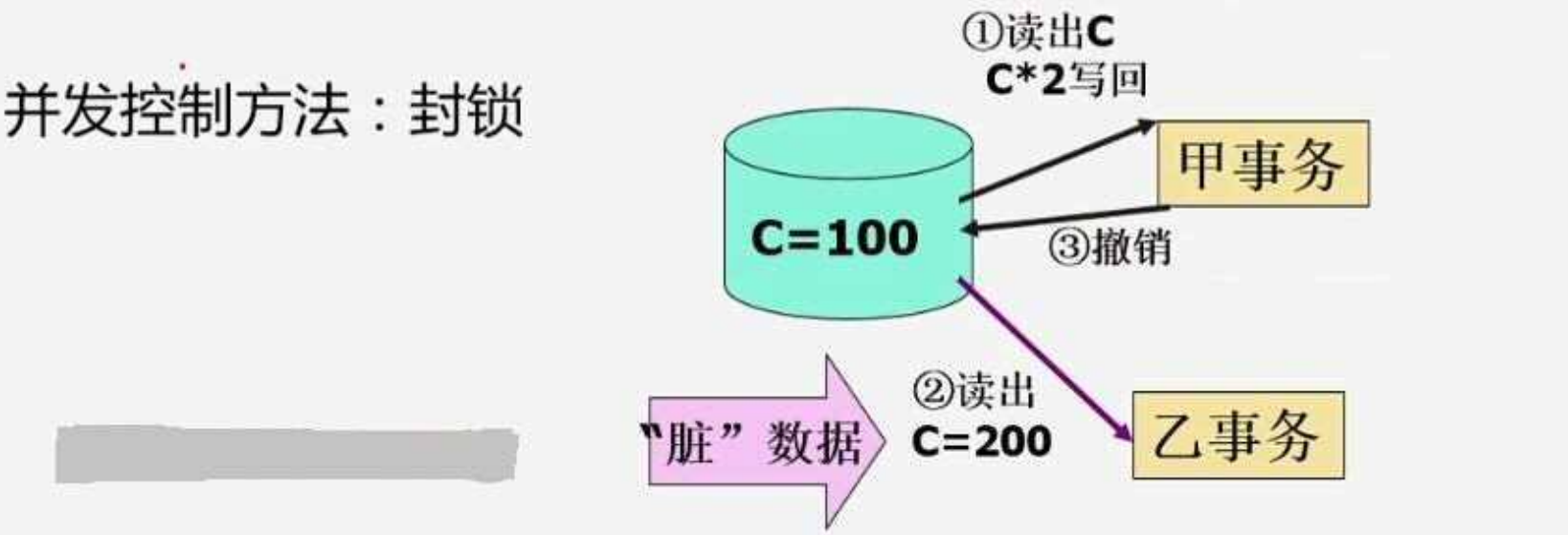

T1读数据后，T2执行更新操作，使T1无法再现前一次读取结果。
①读出B=100
甲事务
B=100
③读不出B
②删除B
乙事务

事务

读脏数据:

T1修改数据并写回磁盘，T2读取同一数据后，T1被撤销即数据恢复原值，T2读的数据与DB中的不一致，称为“脏”数据。

并发控制方法：封锁



事务

封锁

封锁就是事务在对某个数据对象(例如表、记录等)操作之前，先向系统发出请求，对其加锁，加锁后事务就对该数据对象有了一定的控制，在事务释放它的锁之前，其它的事务不能更新此数据对象。

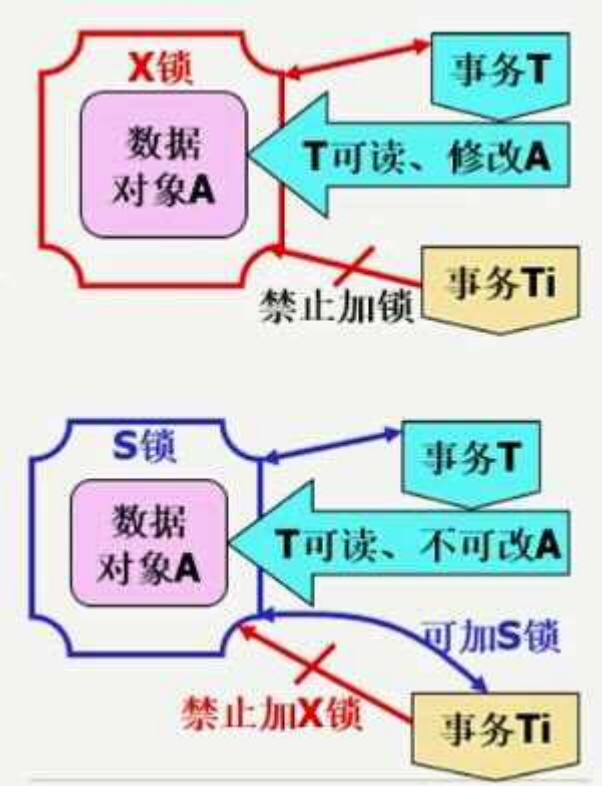
事务

排它锁(X锁或写锁)

保证其它事务在T释放A上的锁之前，不能再修改A。

共享锁(S锁或读锁)

保证其它事务可以读A，但在T释放A上的S锁之前，不能对A做任何修改。



事务

封锁协议

在运用X锁和S锁对数据对象加锁时，还需要约定一些规则，例如何时申请X锁或S锁、持锁时间、何时释放等。称这些规则为封锁协议。对封锁方式规定不同的规则，就形成了各种不同的封锁协议。

一级封锁协议
二级封锁协议
三级封锁协议
两段锁协议

事务

(1)一级封锁协议

事务T在修改数据R之前必须先对其加X封锁，直到事务结束才释放。一级封锁协议可防止丢失修改，并保证事务T是可恢复的。

在一级封锁协议中，如果仅仅是读数据不对其进行修改，是不需要加锁的，所以它不能保证可重复读和不读“脏”数据。

事务

(2) 二级封锁协议

在一级封锁的基础上，事务读数据的时候加s锁，读取之后释放。二级封锁协议可以防止丢失更新，读脏数据。在二级封锁协议中，由于读完数据后即可释放s锁，所以它不能保证可重复读。

Neusoft

事务

(3) 三级封锁协议

一级封锁协议加上事务T在读取数据R之前先对其加S锁，直到事务结束才释放。三级封锁协议可防止丢失修改、防止读“脏”数据，还进一步防止了不可重复读。

T_1	T_2
① Xlock A	
获得	
② 读 A=16	
	Xlock A
③A←A-1	等待
写回 A=15	等待
Commit	等待
Unlock A	等待
④	获得 Xlock A
	读 A=15
	A←A-1
⑤	写回 A=14
	Commit
	Unlock A

(a) 没有丢失修改

T_1	T_2
①Slock A	
Slock B	
读 A=50	
读 B=100	
求和=150	
②	Xlock B
	等待
	等待
③读 A=50	等待
读 B=100	等待
求和=150	等待
Commit	等待
Unlock A	等待
Unlock B	等待
④	获得 Xlock B
	读 B=100
	B←B*2
⑤	写回 B=200
	Commit
	Unlock B

(b) 可重复读

T_1	T_2
①Xlock C	
读 C=100	
C←C*2	
写回 C=200	
②	Slock C
	等待
③ROLLBACK	等待
(C 恢复为 100)	等待
Unlock C	等待
④	获得 Slock C
	读 C=100
⑤	Commit C
	Unlock C

(c) 不读“脏”数据

事务

(4)两段锁协议

所有事务必须分两个阶段对数据项加锁和解锁。其中扩展阶段是在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；收缩阶段是在释放一个封锁之后，事务不能再申请和获得任何其他封锁。

Lock A, Read A, A:=A+100, Write A, Lock B, Unlock A, Read B, Unlock B, Commit

事务

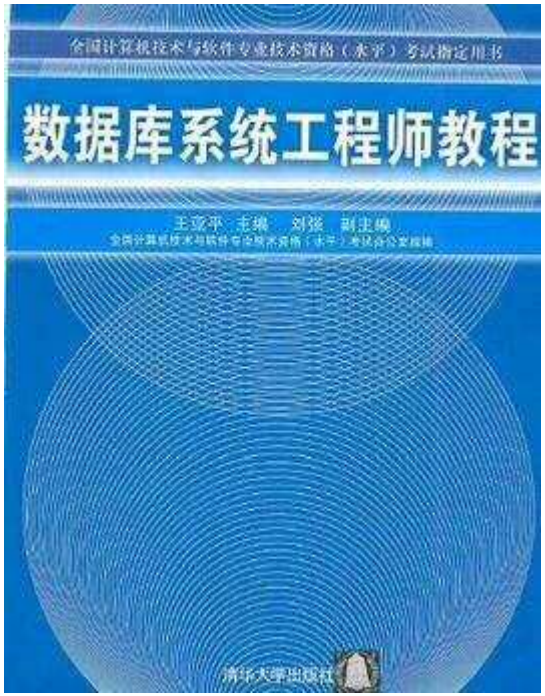
封锁的粒度

封锁的粒度是被封锁数据目标的大小，在关系数据库中封锁粒度有属性值、属性值集、元组、关系、某索引项、整个关系数据库等等。

封锁粒度小则并发性高，但开销大；封锁粒度大则并发性低，但开销小。

参考资料

数据库系统工程师教程




Neusoft
Beyond Technology