

120L020322 刘祚甫

1. (1) R有 $2000/20 = 100$ 块S有 $6000/30 = 200$ 块

将R存入内存需 25 次

则需 $100 + 25 \times 200 = 5100$ 次

(2) R以B排序好

第一次排A: $2 \times 1000 = 2000$

第二次为 3000 次

故一共需 5000 次

(3) $\frac{30}{2} = 15 < 16.6 \approx \frac{20}{1.2}$

一块可存 15 个 R 的 S 组

B为S在R的外键

R 与 S 共有 2000 个元组

有 $\lceil \frac{2000}{15} \rceil = 1667$ 块2. $B(R) = 50$ $B(S) = 50$

$$(1) \therefore \text{I/O 代价} = B(R) + T(R) \lceil \frac{B(S)}{V(S,Y)} \rceil$$

$$= 3050$$

$$(2) \text{I/O 代价} = B(R) + \frac{T(R) \cdot T(S)}{V(S,Y)}$$

$$= 75050$$

3. (1) 前:

 $\pi_{SNAME}(\sigma_{SC.S\# = S.S\# \wedge SC.C\# = C.C\# \wedge CNAME = \text{数据库}})$

数据库

 $(S \bowtie SC \bowtie C)$ (2) (150) π_{SNAME} (150) $\sigma_{SC.S\# = S.S\# \wedge SC.C\# = C.C\# \wedge CNAME = \text{数据库}}$

数据库

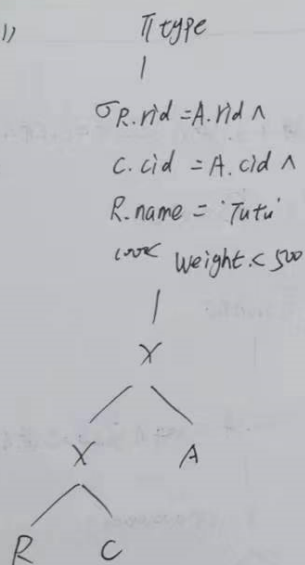
 $X (50000000)$ $(1000000) X C (50)$ $(1000) S SC (1000)$

后:

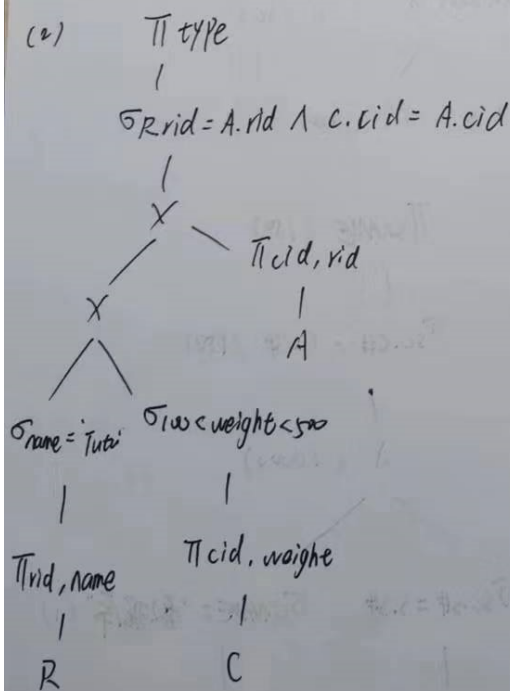
 $\pi_{SNAME} (150)$ $\sigma_{SC.C\# = C.C\#} (150)$ $X (10000)$ $(10000) \sigma_{SC.S\# = S.S\#} \quad \sigma_{CNAME = \text{数据库}} (1)$ $(1000000) X \quad \pi_{C\#, CNAME} (50)$ $\pi_{S\#, SNAME}$ $\pi_{S\#, C\#}$ $C (50)$ $S (1000)$ $SC (10000)$

(3) 计算结果在括号中

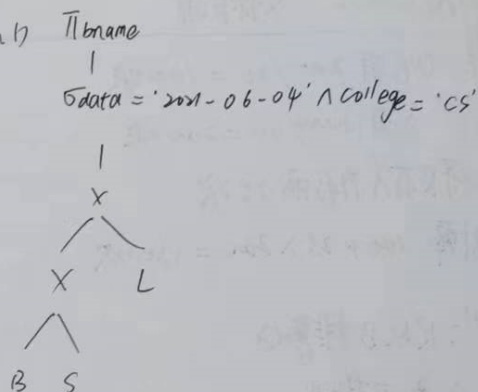
4. (1)



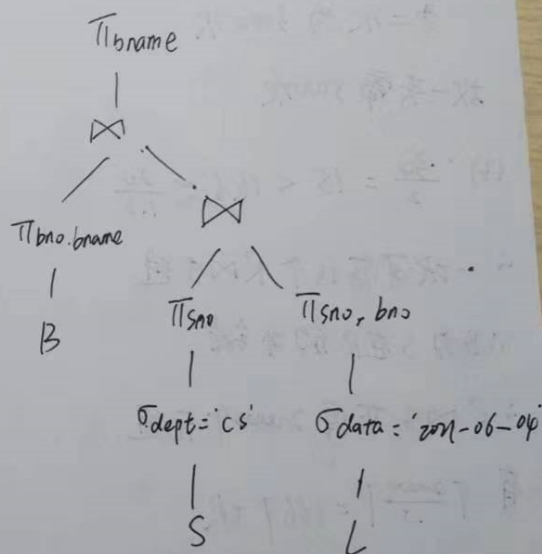
(2)



5. (1)



(2)



理由:

图书远比人数和借书多, 所以最后连接, 按投影 > 选择 > 连接的顺序进行优化

202020322 刘作甫

6. 对于第一个, 可形成 $1 \rightarrow 2 \rightarrow 3$ 循环
 故是可串行化的
 对于第2个,
 $\neg R_1(B), R_2(B), W_1(B), W_2(B)$
 $\therefore 1 \not\sim 2$, 不可串行

7. (1)
 两阶段封锁协议: 读或写数据前要获得锁,
 每个事务中所有封锁请求先于任何一个解
 锁请求

(2) 两阶段: 加锁段, 解锁段, 加锁段
 中不可有解锁操作. 解锁段中不可有
 加锁操作

T_1 :	$L_1(A)$	T_2 :	
	$L_1(B)$		$L_2(B)$
	$read(A, t_1)$		$L_2(A)$
	$read(B, t_2)$		$read(B, t_1)$
	$if\ t_1 > t_2$:		$read(A, t_2)$
	$t_2 = t_1$		$if\ t_1 < 0$
	$while(B, t_1)$		$t_2 = t_1 \times t_1$
	$U_1(A)$		write
	$U_2(B)$		$write(A, t_2)$
			$U_2(B)$
			$U_2(A)$

T_1	T_2
$L_1(A), read(A, t_1)$	$L_2(A)$, 拒绝
$if(t_1 > t_2): t_2 = t_1$	$L_2(B)$, 拒绝
$write(B, t_2)$	获得锁:
$U_2(B), U_1(A)$	$read(B, t_3)$
	$read(A, t_4)$
	$if(t_3 < 0): t_4 = t_3 \times t_3$
	$write(A, t_4)$
	$U_2(A), U_2(B)$

T_1	T_2
$L_1(A), read(A, t_1)$	$L_2(B), read(B, t_3)$
$L_1(B)$, 拒绝	$L_2(A)$, 拒绝

(4) 等待图: 将事务之间的等待图关系表示
 为一个有向图, 若这个图存在
 环路, 则说明发生死锁

超时机制: 每个等待资源的事务设置个
 超时时间, 若超时则主动
 放弃并将资源回滚

8. (1) 检查点: 在日志文件中增加检查点
WAL机制: 在执行更改数据前, 先将日志写入日志文件

undo/redo: 在修改数据时, 改变前后的数据都需要记录

(2) T_1, T_3 需要 undo, T_2 需要 redo,
因为三者均有 start, 但只有 T_2 commit.

(3) A: 114514, B: hits

过程: 先正向扫描

$T_1 \rightarrow \text{undo} \rightarrow \text{list}$

A: 114 \rightarrow 114514

$T_2 = \text{undo} - \text{list}$

B: hit \rightarrow hits

$\text{undo} - \text{list}, \text{delete}(T_1)$

$T_3 \rightarrow \text{undo} - \text{list}$

B: hits \rightarrow hitsdb

A: 114514 \rightarrow 1919810

再反向扫描:

A: 1919810 \rightarrow 114514

B: hitsdb \rightarrow hits

9.

(1) T_1 不用操作

T_2, T_4 : redo

T_3, T_5 : undo

(2) redo: T_6

undo: T_7, T_8

(3) 正向:

$\text{undo-list.add}(T_6)$

X: 100 \rightarrow 1

$\text{undolist.add}(T_1)$

X: 1 \rightarrow 3

$\text{undolist.add}(T_8)$

Y: 6 \rightarrow 8

Z: 10 \rightarrow 9

$\text{undolist.delete}(T_6)$

Z: 9 \rightarrow 10

反向:

Z: 10 \rightarrow 9

Z: 9 \rightarrow 10

Y: 8 \rightarrow 6

Y: 6 \rightarrow 50

X: 3 \rightarrow 1