1. 已知 2 个关系 R(A, B)和 S(B, C)，其主键分别为 R.A 和 S.B。R 有 20000 个元组，S 有 60000 个元组，一块中可以容纳 20 个 R 元组或 30 个 S 元组。设 2 个关系均采用聚簇存储，且每个关系中的元组均已按照其主键值递增排序。现在要执行自然连接操作 R ⋈ S。设缓冲区中可用内存页数为 M = 41。回答下列问题：

(1) 采用嵌套循环连接算法执行 R ⋈ S 分别需要进行多少次 I/O？给出具体分析过程。

答:

①基于元组的嵌套循环连接:

若选择 R 作为外部关系，I/O 代价:T(R)(T(S)+1)=20000*(60000+1)=1200020000 次

若选择 S 作为外部关系，I/O 代价:T(S)(T(R)+1)=60000*(20000+1)=1200060000 次

②基于块的嵌套循环连接:

B(R)=20000/20=1000，B(S)=60000/30=2000，有 B(R)<B(S)

I/O 代价：B(R)+B(S)*$\frac{B(R)}{M-1}$ =1000+2000*25=51000 次

(2) 采用归并连接算法执行 R ⋈ S 分别需要进行多少次 I/O？给出具体分析过程。

答:

根据题意“每个关系中的元组均已按照其主键值递增排序”的 R 已经按照 R.A 排序，S 已经按照 S.B 排序。在归并连接算法中，首先需要进行排序，由于 S 已经排过序，因此不再需要此操作；对于 R，创建归并段需要 B(R)次 I/O，将 R 的归并段写入文件，需要 B(R)次 I/O。在归并阶段，对 R 和 S 的每个归并段各扫描 1 次,合计 B(R) +B(S)次 I/O。

合计 I/O 次数:3B(R)+B(S)=5000 次。

(3) 设 R.B 是关系 R 的外键，参照 S.B。如果 R ⋈ S 的结果中元组的平均大小是 R 中元组平均大小的 1.2 倍，R ⋈ S 的结果中元组的平均大小是 S 中元组平均大小的 2 倍，那么在外存中存储 R ⋈ S 的结果需要占用多少个块（页）？给出具体分析过程。

答:

因为 R.B 是关系 R 的外键，所以 R ⋈ S 的结果中包含的元组数一定与 R 的元组数相同。

因为 R ⋈ S 的结果中元组的平均大小是 R 的 1.2 倍，则 1.2*（20000/20）=1200 个块。

2. 设关系 R(X, Y)和 S(Y, Z)，R 共有 1000 个元组，S 共有 1500 个元组，每个块中可容纳 20 个 R 元组或 30 个 S 元组。S 中 Y 不同值的个数为 20。

B(R)=1000/20=50,B(S)=1500/30=50,V(S,Y)=20

(1) 若在 S.Y 上建有聚簇索引，估计 R 和 S 基于索引连接的 IO 代价。

答：因为索引是聚簇索引,所以对于 R 的每个元组 r,S 中能与 r 连接的元组一定连续存储于 S 的文件中,约占⌈B(S)/V(S,Y) ⌉个块

合计 I/O 次数：50+1000*3=3050 次

(2) 若在 S.Y 上建有非聚簇索引，估计 R 和 S 基于索引连接的 IO 代价。

答：R 的每块只读 1 次，合计 B(R)次 I/O; 对于 R 的每个元组 r, S 中平均约有(T(S))/(V(S, Y))个元组能与 r 连接；因为索引是非聚簇索引,这些元组在文件中不一定连续存储。最坏情况下,读每个元组产生 1 次 I/0,合计 T(R)T(S)/V(S,Y) 次 I/0

合计 IO 次数：50+1000*1500/20=75050 次

3. 设教学管理数据库有如下 3 个关系模式：

S(S#, SNAME, AGE, SEX)

C(C#, CNAME, TEACHER)

SC(S#, C#, GRADE)

其中 S 为学生信息表、SC 为选课表、C 为课程信息表；S#、C#分别为 S、C 表的主码，(S#, C#)是 SC 表的主码，也分别是参照 S、C 表的外码

用户有一查询语句：

Select SNAME

From S, SC, C

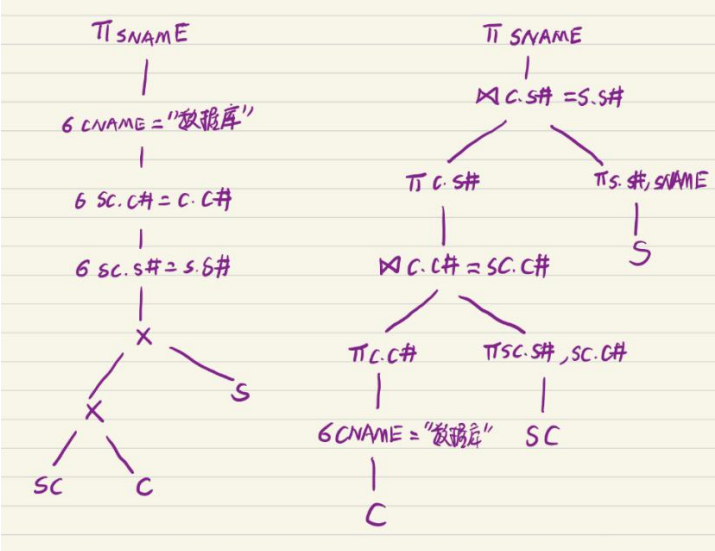
Where SC.S#=S.S# and SC.C#=C.C# and CNAME= “数据库”

检索选学“数据库”课程的学生的姓名。

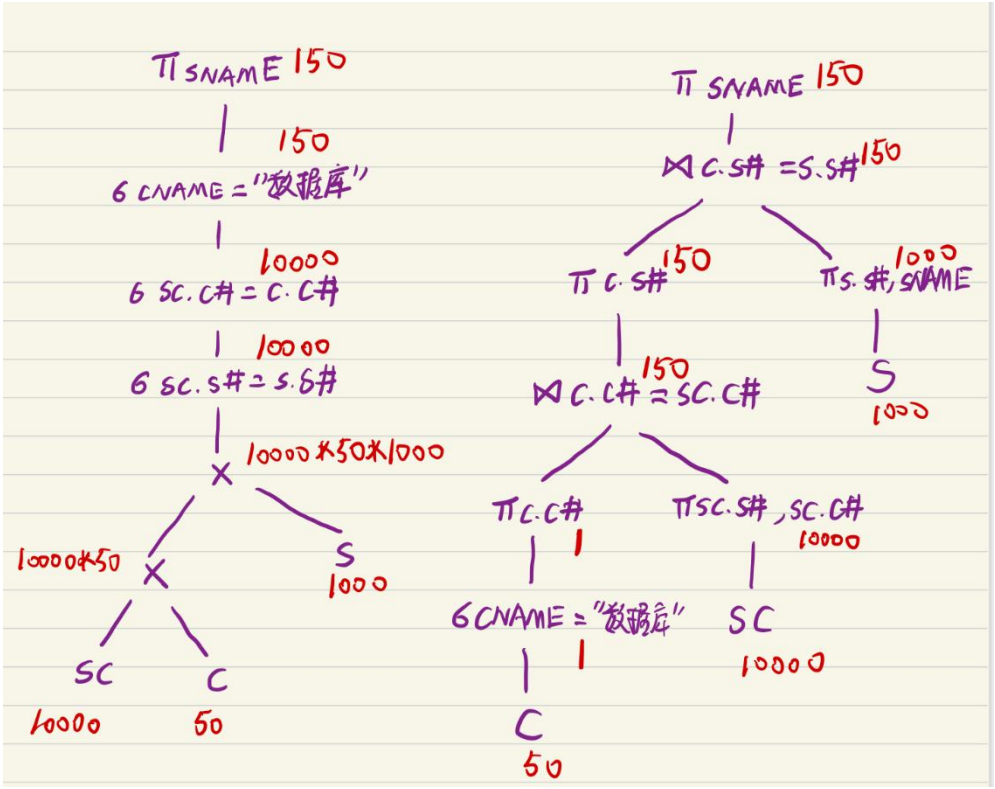
(1)写出以上 SQL 语句所对应的关系代数表达式。

$\Pi_{SNAME}(\sigma_{SC.S\#=S.S\# \wedge SC.C\#=C.C\# \wedge CNAME='数据库'}(S \times SC \times C))$

(2)画出上述关系代数表达式所对应的查询计划树。使用启发式查询优化算法，对以上查询计划树进行优化，并画出优化后的查询计划树。



(3)设 SC 表有 10000 条元组，C 表有 50 条元组，S 表中有 1000 条元组，SC 中满足选修数据库课程的元组数为 150，计算优化前与优化后的查询计划中每一步所产生的中间结果大小

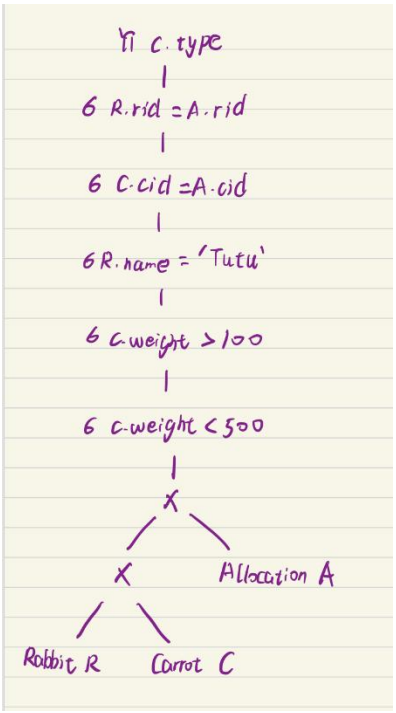


4. 给定以下关系模式，

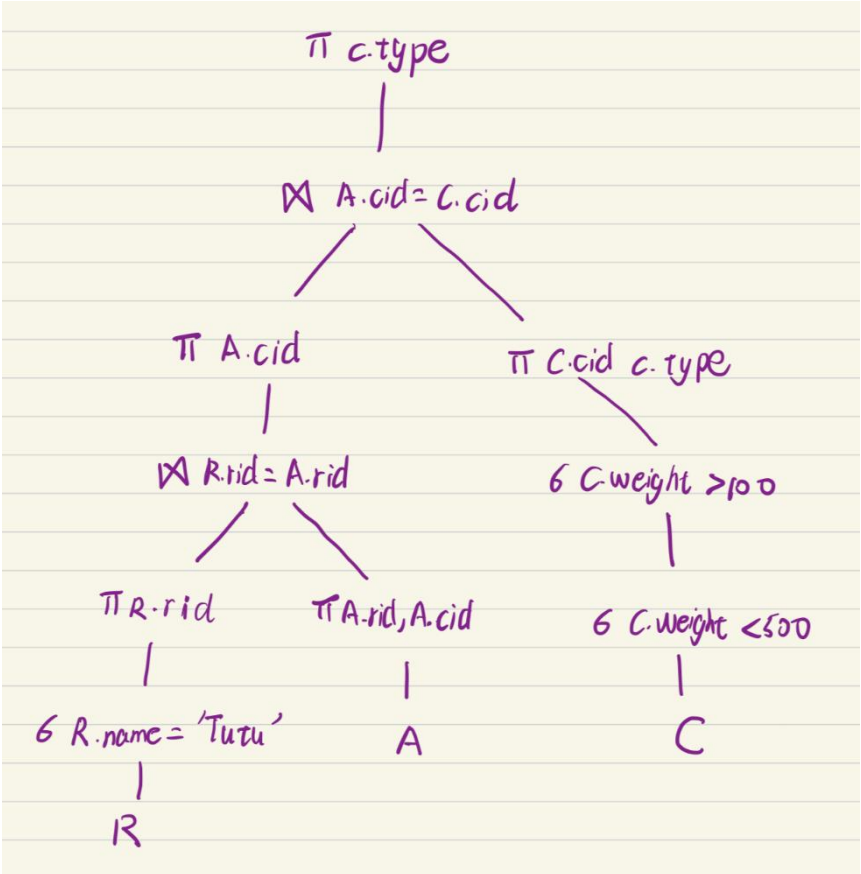
```
Rabbit (rid, name, color)
Carrot (cid, weight, type)
Allocation (rid, cid, date)
```

(1) 考虑以下的 SQL 查询语句，绘制其查询计划树。

```
SELECT C.type
FROM Rabbit R, Carrot C, Allocation A
WHERE R.rid = A.rid
AND C.cid = A.cid
AND R.name = 'Tutu'
AND C.weight > 100
AND C.weight < 500
```



(2) 假设在 Rabbit.name 和 Allocation.rid 上建有索引，绘制优化后的查询计划树。

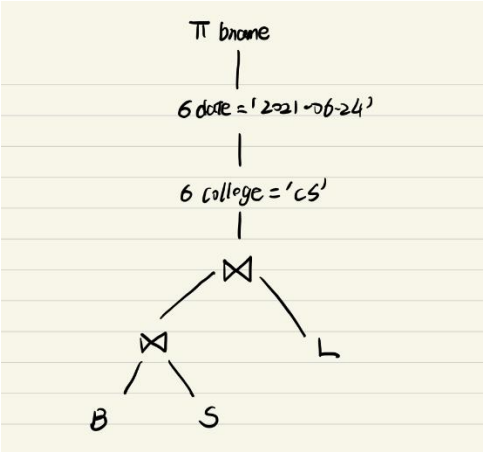


5. 已知一个关系数据库的模式如下：

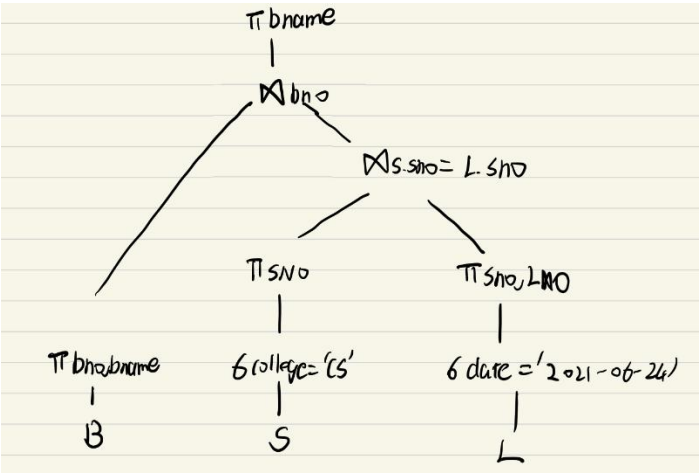
关系 B(bno, bname, author)为图书表，其中 bno 为书号，bname 为书名，author 为作者；
关系 S(sno, sname, dept)为学生表，其中 sno 为学号，sname 为姓名，dept 为学生所在系；
关系 L(sno, bno, date)为借书表，其中 sno 为学号，bno 为书号，date 为借书时间。
回答下列问题：

(1) 绘制下面的 SQL 查询语句的逻辑查询计划树。

```
SELECT bname FROM B NATURAL JOIN S NATURAL JOIN L
WHERE date = '2021-06-04' AND college = 'CS';
```



(2) 使用启发式查询优化方法对上面的逻辑查询计划树进行优化，绘制优化后得到的逻辑查询计划树，具体说明你进行这些优化的理由。
主要的优化有：将投影和选择下移至接近叶结点处，这样在进行连接操作时的代价较小



6. 设 ri(X)与 wi(X)分别表示事物 Ti 读、写数据单元 X，则一个并发调度可以抽象为读、写串。基于上述表示，请判断下面两个并发调度是否是可串行化的，为什么？

调度 S：

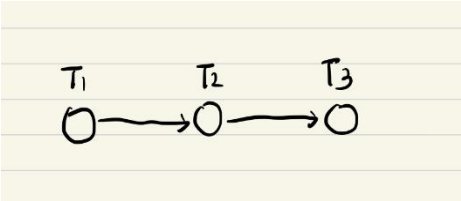
$$r_2(A); r_1(B); w_2(A); r_3(A); w_1(B); w_3(A); r_2(B); w_2(B)$$

调度 S’：

$$r_2(A);r_1(B);w_2(A);r_2(B);r_3(A);w_1(B);w_3(A);w_2(B)$$

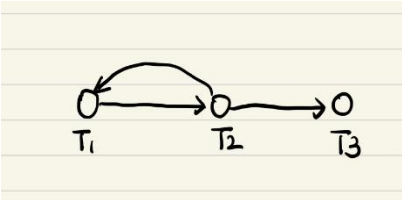
答：

S 是可串行化的，画出 S 对应的优先图：



无环，说明其是可串行化的。

S’不是可串行化的，画出 S’对应的优先图：



存在环，因此不是可穿行化的。

7. 已知下面两个事务：

```
T1:
read(A);
read(B);
if A > B then B := A;
write(B);


T2:
read(B);
read(A);
if B < 0 then A := B * B;
write(A);
```

回答下列问题：

(1) 简述两段锁协议（简称 2PL），并给事务 T1 和 T2 添加加锁和解锁指令，使其遵从 2PL 协议。

两段锁协议: 两段锁协议（2PL）是指事务在执行过程中，必须先获得所有需要用到的锁，直到事务结束时才释放所有锁。2PL 由两个阶段组成，分别是加锁阶段和解锁阶段。加锁阶段：事务在读取或修改数据前必须先获得相应

的锁；解锁阶段：事务完成所有操作后，释放所有锁。在 2PL 中，加锁和解锁的顺序必须一致，即先加锁再解锁。

T1:
LOCK-S(A)
read(A)
LOCK-X(B)
if A > B then B := A;
write(B);
UNLOCK(A)
UNLOCK(B)

T2:
LOCK-S(B)
read(B);
LOCK-X(A)
read(A);
if B < 0 then A := B * B;
write(A);
UNLOCK(A)
UNLOCK(B)

(2) 按照你添加的加锁和解锁指令，给出 2PL 下对 T1 和 T2 的一个正确的调度.

T1	
LOCK-S(A)	
read(A)	
LOCK-X(B)	
if A > B then B := A;	
write(B);	
UNLOCK(A)	
UNLOCK(B)	
	LOCK-S(B)
	read(B);
	LOCK-X(A)
	read(A);
	if B < 0 then A := B * B;
	write(A);
	UNLOCK(A)
	UNLOCK(B)

(3) 什么情况下，T1 和 T2 会发生死锁？请给出一个会发生死锁的调度。

当两个或多个事务都已封锁了一些数据对象，然后又都请求对已为其他事务封锁的数据对象加锁，从而出现死等待。

下面这种调度就会发生死锁：

T1	T2
LOCK-S(A)	
read(A)	
	LOCK-S(B)
	read(B);
LOCK-X(B)	
	LOCK-X(A)

(4) 在数据库的实际应用中经常会遇到的与锁相关的异常情况，请简述一下常见的死锁检测方法

①超时检测：这种方法是最简单的死锁检测方法之一。它基于一个假设：如果一个事务等待锁的时间超过了一个预设的时间限制，那么它可能已经发生了死锁。在这种情况下，系统会中断其中一个事务，以打破死锁。

②等待图算法：这种方法使用一个称为“等待图”的数据结构来表示事务之间的依赖关系。等待图中的节点表示事务，边表示事务之间的依赖关系。当一个事务等待另一个事务所持有的锁时，等待图中就会出现一条边。如果等待图中存在环路，那么系统就会检测到死锁。

8. 已知某数据库采用即时更新方法（undo-redo 方法）记录 WAL 日志。设故障发生时 WAL 日志文件内容如下：

```
<T1, begin>
<T1, A, 114, 114514>
<T2, begin>
<T1, B, “hit”, “hitcs”>
<T1, commit>
<T3, begin>
<T3, B, “hitcs”, “hitcsdb”>
<T2, A, 114514, 1919810>
```

当系统重启后，DBMS 基于该 WAL 日志文件进行故障恢复。回答下列问题：

(1)该数据库系统的的日志恢复策略为 Undo/Redo 型，那么其对应的缓冲区处理策略是什么？该策略的每一项的具体内容都有什么？

答：

1. 所有 〈T, X, v1,v2〉 型日志记录安全地、永久存储到存储器之前，事务 T 不能更新数据库。当 T 发出一个 write(X) 操作时，记录 〈T, X, v1,v2〉 首先被写入日志，然后直接在数据库上执行 write(X)。

2. 所有〈T, X, v1, v2〉型日志记录安全地、永久存储到存储器之前，不允许事务 T 提交。

(2)当 DBMS 进行故障恢复时，需要对那些事务进行 undo？对那些事务进行 redo？给出具体

答：故障恢复时，由于 T1 已经 commit，T2, T3begin 但未 commit，因此对 T3, T2 进行 undo，对 T1 进行 redo。

undo:

<T3,B,"hitcs","hitcsdb">

<T2,A,114514,1919810>

redo:

<T1,A,114,114514>

<T1,B,"hit","hitcs">

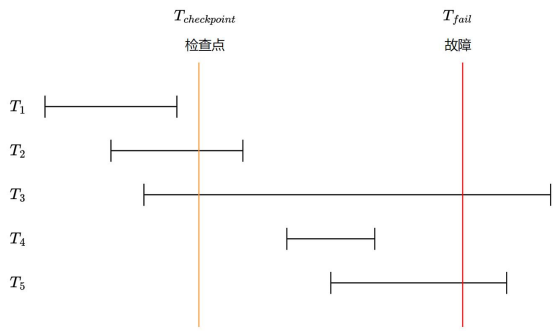
(3) 当故障恢复完成时，对象 A 和 B 的值分别是什么？描述故障恢复的具体过程。

故障恢复的具体过程如下：

从后向前扫描日志记录，建立两个事务表：一个表称为提交事务表，包含 T1；另一个表称为未提交事务表，包含 T2, T3，对于未提交事务表中的每个事务 T（T2, T3），执行 undo(T)，写一个 <T,abort> 日志记录，表明撤销完成；对于提交事务表中的每个事务 T（T1），执行 redo(T)。

由于 T1 事务已经提交，因此系统首先会执行 T1 的 redo 操作，将对象 A 和 B 的值分别修改为 114514 和“hitcs”。接下来，系统会执行 T2 和 T3 的 undo 操作，将对象 A 和 B 的值分别修改为 114514 和“hitcs”。因此，在故障恢复完成后，对象 A 的值为 114514，对象 B 的值为“hitcs”。

9. 使用检查点的数据库恢复系统将根据事务的状态和检查点的关系采取相对应的恢复策略，现在有事务T_1-T_5其执行过程如下图所示（线段左端和右端分别表示事务开始与提交），其执行过程中数据库系统发生如图所示的故障，请回答下列问题



(1)请问在故障恢复时事务T_1-T_5那些需要撤销，那些需要重做，那些不需要操作？

需要撤销的：T3,T5

需要重做的：T2,T4
不需要操作的：T1
(2)事务T_6-T_8 的日志文件如下图所示,<T_i，begin>表示事务T_i开始执行，<T_i，commit>表示事务T_i提交，<T_i，D，V_1，V_2>表示事务T_i 将数据项 D 的值由V_1修改为V_2，<crash>表示数据库发生故障

<T_6, begin>
<T_6, X, 100, 1>
<T_7, begin>
<T_7, X, 1,3>
<T_8, begin>
<T_7, Y, 50, 6>
<T_8, Y, 6, 8>
<T_8, Z, 10, 9>
<checkpoint>
<T_6,commit>
<T_8,Z,9,10>
<crash>

数据库系统发生故障时，请给出恢复子系统时需要 undo 的事务列表和需要 redo 的事务列表
undo：T7,T8
<T_7, X, 1,3>
<T_7, Y, 50, 6>
<T_8, Y, 6, 8>
<T_8, Z, 10, 9>
<T_8,Z,9,10>
redo：T6
<T6,X,100,1>
(3)请简述事务T_6-T_8 在系统故障后，基于检查点的故障恢复过程
系统检查日志以找到最后一条<checkpoint L>记录，对 L 中的事务，以及<checkpoint L>记录写到日志中之后才开始执行的事务，做如下操作：
对于满足上述要求的 T，若日志中没有<T, commit>或<T, abort>记录，则执行 undo(T)，否则执行 redo(T)。在 checkpoint 后，仅 T6 存在 commit 记录，因此执行 redo 操作，对 T7,T8 执行 undo 操作。