1.在$R(k,a)$和$S(k,b)$两个关系上，有下面两个查询计划 A 和 B

Plan A: $(\sigma_{a=3}R) \bowtie S$
Plan B: $\sigma_{a=3}(R \bowtie S)$
回答下面两个问题：
a.如果没有R和S上属性值分布的知识，一般情况下你选择哪个执行计划，为什么？
b.是否存在一种情况你会选择另外一个计划，为什么？

2.考虑如下的三个事务的调度。这是否是一个冲突串行化调度？请解释一下你的判据。如果是，将它转换成串行调度。

T0	T1	T2
r0(A)		
w0(A)		
		r2(A)
		w2(A)
	r1(A)	
r0(B)		
		r2(B)
w0(B)		
		W2(B)
	r1(B)	

3.用两阶段锁协议保证题目 2 的调度的冲突可串行化（写出加锁和去锁的过程，可以附上必要的文字说明）

4.考虑两个事务 T1，T2。其中，T1 显示账户 A 与 B 的内容：
T1: Read(B);
Read(A);
Display(A+B).

T2 表示从账户 B 转 50 美元到账户 A，然后显示两个账户的内容：
T2: Read(B);
B := B-50;
Write(B);
Read(A);
A := A+50;
Write(A);
Display(A+B).
令 TS（T1），TS（T2）分别是事务 T1 和 T2 开始的时间戳，并且 TS（T2）<TS（T1）。现有如下一个调度。请给出每一步骤中时间戳 W-ts(A)，R-ts(A)，W-ts(B)，R-ts (B)的取值。(注：Display(A+B)表示显示账户 A 和账户 B 的内容，

时间戳若不可知则留为空）

T1	T2	W-ts(A)	R-ts(A)	W-ts(B)	R-ts(B)
	Read(B);				
	B := B-50;				
	Write(B);				
Read(B);					
	Read(A);				
	A := A+50;				
Read(A);					
	Write(A);				
Display(A+B).					
	Display(A+B).				

5.设一个数据库系统启动后中，执行 4 个事务 T0、T1、T2 和 T3。四个事务的内容如下：

T0: A := A + 20 (读入数据库元素 A 的值，加上 20 后，再写回 A 的值)

T1: B := B - 10 (读入数据库元素 B 的值，减去 10 后，再写回 B 的值)

T2: C := C * 2 (读入数据库元素 C 的值，乘以 2 后，再写回 C 的值)

T3: D := D + 15 (读入数据库元素 D 的值，加上 15 后，再写回 D 的值)

除了这四个事务外,系统中无其他事务执行。设四个事务开始前,数据库元素 A、B、C、D 的值分别为 A = 50，B = 30，C = 35，D = 15。在执行这四个事务的过程中，系统发生了故障。系统重启后，经故障恢复，数据库元素 A、B、C、D 的值被恢复为 A = 50，B = 20，C = 70，D = 15。故障恢复时，数据库系统日志文件中包含如下 12 条日志记录，这里只给出部分日志记录。已知该数据库管理系统使用基于 undo-redo 日志的故障恢复技术,这段日志中仅有 1 个不停机检查点(又称模糊检查点)。

1	
2	<T0, A, 50, 70>
3	
4	<start checkpoint (T0, T2)>
5	<end checkpoint>
6	<T1, start>
7	
8	<T1, commit>
9	<T2, C, 35, 70>
10	
11	<T3, D, 15, 30>
12	

请根据上述信息，回答下列问题：

a.将日志文件补充完整，直接在上面的日志文件中填写。一个事务 T 启动时向日志文件中写入日志记录<T, start>；提交时向日志文件中写入日志记录<T, commit>；中止时向日志文件中写入日志记录<T, abort>；对数据库元素 X 进行

修改时向日志文件中写入日志记录<T，X，X 的旧值，X 的新值>。
b. 在故障恢复过程中，哪些事务需要 redo，哪些事务需要 undo。说明理由。
c. 在故障恢复过程中，还会向日志文件添加什么日志记录？说明理由。

请将作业在 2019 年 5 月 22 日（下周三）晚 10 点之前发送到作业邮箱 hitdatabase2019@163.com。请提交 pdf 或者 word 格式文件（可以手写然后拍照粘贴到文件中），不接收其他格式尤其照片压缩包。文件命名为“学号-姓名-第六次作业”。顺利发送将受到回复。

1. a. A，因为一般情况下选择下推可以大大减小中间结果大小

b. 当下推的选择操作不能减小中间结果或先做连接中间结果规模较小时，可以使用B计划.

2. 不是冲突串行化调度.

由 $Y_0(B)$、$W_2(B)$ 可知，$T_0 < T_2$，优先图中有一条 T_0 指向 T_2 的弧

由 $Y_2(B)$、$W_0(B)$ 可知，$T_2 < T_0$，优先图中有一条 T_2 指向 T_0 的弧

所以优先图中有环，不是冲突串行化调度.

3.

T_0	T_2	T_1
LOCK-X(A)	LOCK-X(A)	LOCK-S(A)
$Y_0(A)$	$Y_2(A)$	$Y_1(A)$
$W_0(A)$	$W_2(A)$	LOCK-S(B)
LOCK-X(B)	LOCK-X(B)	$Y_2(B)$
$Y_0(B)$	$Y_2(B)$	UNLOCK(A)
$W_0(B)$	$W_2(B)$	UNLOCK(B)
UNLOCK(A)	UNLOCK(A)	
UNLOCK(B)	UNLOCK(B)	

4.

T1	T2	W-ts(A)	R-ts(A)	W-ts(B)	R-ts(B)
	Read(B);				TS(T_2)
	B := B-50;				TS(T_2)
	Write(B);			TS(T_2)	TS(T_2)
Read(B);				TS(T_2)	TS(T_1)
	Read(A);		TS(T_2)	TS(T_2)	TS(T_1)
	A := A+50;		TS(T_2)	TS(T_2)	TS(T_1)
Read(A);			TS(T_1)	TS(T_2)	TS(T_1)
	Write(A); 回滚				
Display(A+B).	TS(T_2)<TS(T_1)				
	Display(A+B).				

执行到 T_2 事务的 Write(A) 时，由于 TS(T_2) < R-TS(A)，T_2 回滚，分配新的时间戳 TS'(T_2)，且 TS'(T_2) > TS(T_1)，重新执行 T_2.

回滚后重新执行结果如下：

T2	W-ts(A)	R-ts(A)	W-ts(B)	R-ts(B)
Read(B);		$TS(T_1)$	$TS(T_2)$	$TS'(T_2)$
B := B-50;		$TS(T_1)$	$TS(T_2)$	$TS'(T_2)$
Write(B);		$TS(T_1)$	$TS'(T_2)$	$TS'(T_2)$
Read(A);		$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$
A := A+50;		$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$
Write(A);	$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$
Display(A+B).	$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$	$TS'(T_2)$

5.

1	$<T_0, start>$
2	<T0, A, 50, 70>
3	$<T_2, start>$
4	<start checkpoint (T0, T2)>
5	<end checkpoint>
6	<T1, start>
7	$<T_1, B, 30, 20>$
8	<T1, commit>
9	<T2, C, 35, 70>
10	$<T_3, start>$
11	<T3, D, 15, 30>
12	$<T_2, commit>$

b. redo = T_1, T_2　　undo = T_0, T_3

在检查点处，T_0和T_2活跃，undo = T_0, T_2

从检查点向后搜索，$<T_1, start>$，undo = T_0, T_1, T_2

$<T_1, commit>$，undo = T_0, T_2，redo = T_1

$<T_3, start>$，undo = T_0, T_2, T_3，redo = T_1

$<T_2, commit>$，undo = T_0, T_3，redo = T_1, T_2

C. 由于T_0, T_3需要redo，加入$<T_0, abort>$ $<T_3, abort>$.