以下来自学长版本，不是官方标答，仅供复习参考。

第十三章: Secondary Storage Management

13.2.1

a)磁盘的容量为：

$10\times100000\times1000\times1024B=1.024\times10^{12}B$

b)平均为密度是：每个磁道有 2000*1024*8 位数据，占据磁道的 90%。最外圈的磁道长度是 3.5×3.14 即大约 11 英寸，这个大小的 90%即大约 9.9 英寸。所以磁道所占据部分的位密度大约是每英寸 2000*1024*8/9.9 位，也就是大约 1654949.4949494949495 位 c)最大寻道时间是： 1+0.0003*100000=31 milliseconds

d)最大旋转等待时间是： 60/6000s=10 ms

e)10*(36*(65546/1024-1)/256+324*(65546/1024)/256)/360 ms

f)平均寻道时间是：1+(100000/3)*0.0003

g)平均旋转等待时间是：10÷2=5ms

13.2.2

假设磁头起初以相同的概率被定位在 8192 个柱面的任一位置。如果是在柱面 1 或柱面 8192，那么要移动的平均磁道数（1+2+...+8191）/8192,即大约 4096 磁道，即中间位置，则磁头移进或移出的可能性是相同的，而且无论移进还是移出，移动距离平均来说大约都是总磁道数的四分之一，即 2048 磁道。

当磁道数为偶数时，平均磁道数为：

(((1+n-1)*(n-1)*n/2-(1+n/2-1)*(n-1)*(n-1)+((n/2-1)*(n/2)*(n-1))/6-1-((1+(n/2))*((n/2)-2))/2)*2)))/((n-1)(n-1))约等于 n/3

当磁道数为奇数时，证明过程同上，略

13.3.1

a)采用电梯算法

请求的柱面	完成时间
8000	11.3ms
4000	17.6ms
40000	31.9ms
48000	39.2ms

b)采用先到达先服务

请求的柱面	完成时间
8000	11.3ms
48000	26.6ms
4000	42.9ms
40000	57.2ms

13.3.4

假设一个柱面的扇区数为 x，盘片数为 y,则平均访问的扇区为/（x*y）

13.4.2

10

00

10

13.4.3

$$\frac{1}{20}\times\frac{1}{876}=\frac{1}{17520}$$

所以，磁盘平均故障时间为 10×17520=175200 年

13.4.5

a)00000110　　b)01110101

13.4.7

a)01101110　　b)10111010

13.5.1

a)23+2+10+10=45　　b)24+8+16+16=64　　c)24+4+12+12=52

13.5.2

a)4+4+1+45=54　　b)8+8+8+64=88　　c)4+4+4+52=64

13.6.5

747 中有 2 的 28 次方个扇区，2 的 14 次方大于 10000，所以总共需要 28+14=42 位，也就是需要 6 个字节来表示块地址。

13.6.7

$P>\frac{1}{2}$

13.6.9

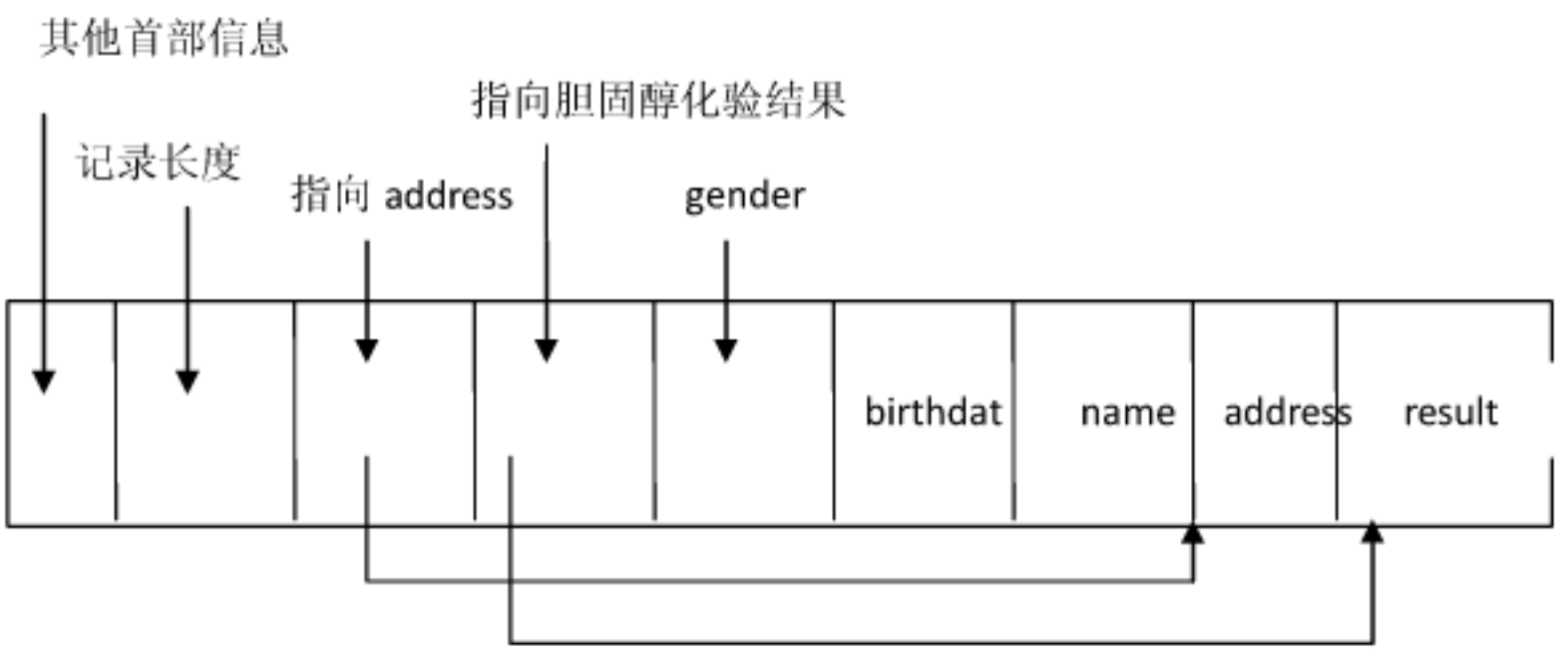
由 $\begin{cases}(x+1)\times 200+4x\leqslant 4096\\(x+2)\times 200+4(x+1)\geqslant 4096\end{cases}$

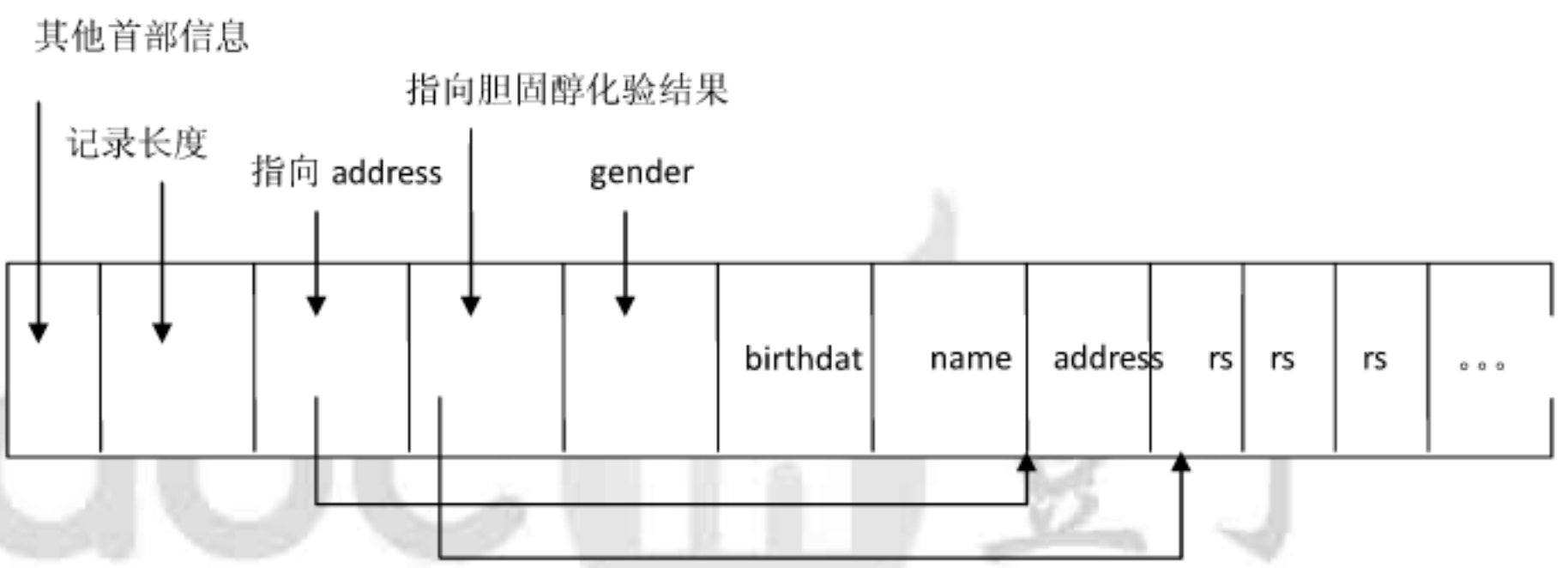
得　x=19,所以 19 天后不再有插入记录的空间

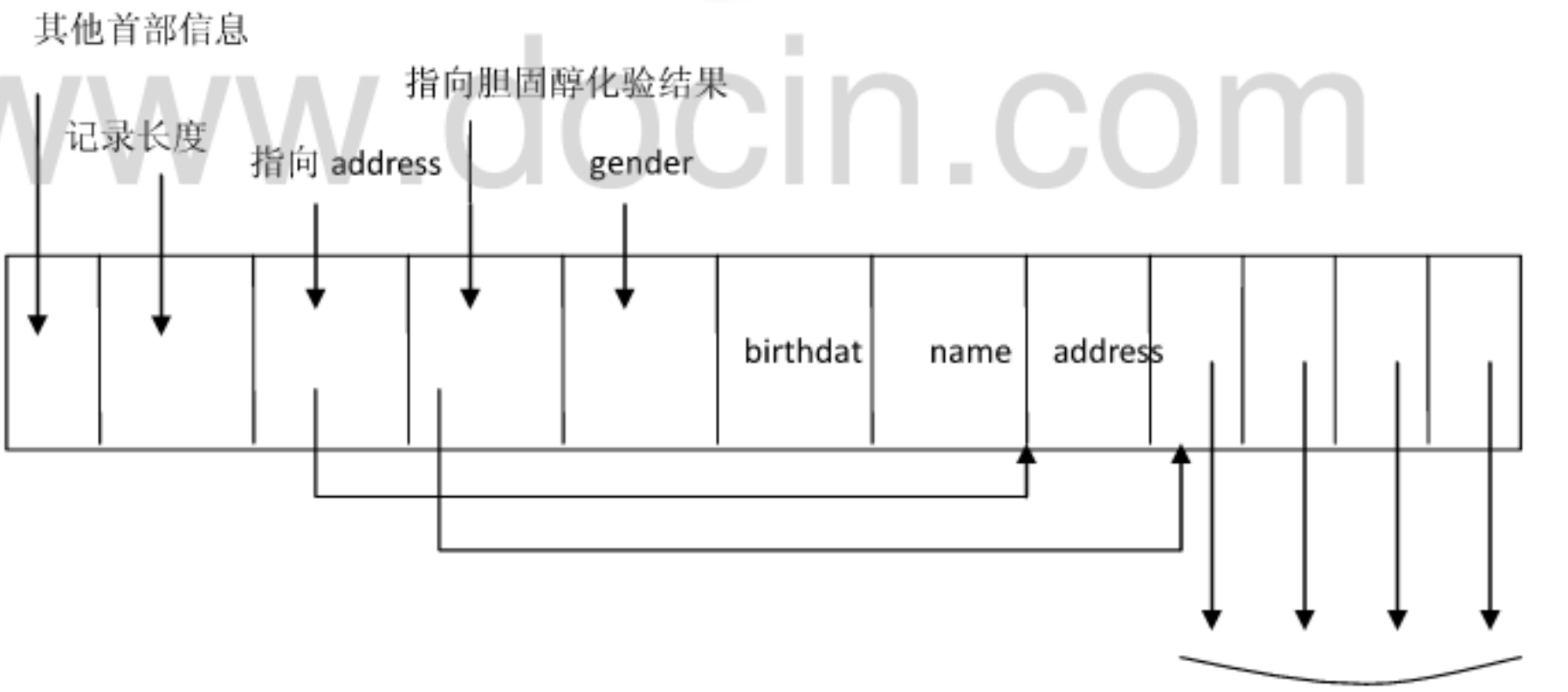
13.7.1

3*9+3*8+2=53

13.7.3







13.8.1

1）使用定长记录有利于 dbms 检索记录

2）有利于 dmbs 修改记录（无需考虑修改后的记录比以前的记录大还是小）

3）有利于 dmbs 对记录的某个字段做索引

第六章：系统对策故障

6.1.1

a)不保持一致性
b)保持一致性
c)保持一致性

6.2.3

a)A 恢复为 10，<ABORT　U>，<ABORT　T>
b)C 恢复为 30，<ABORT　U>
c)E 恢复为 50，C 恢复为 30，<ABORT　U>
d)什么都不做

6.2.7

a)
　　i)<COMMIT　S>之后
　　ii)
b)
　　i)<COMMIT　T>之后
　　ii)
c)　i) <COMMIT　T>之后
　　ii)
d)　i) <COMMIT　V>之后
　　ii)
e)　i) <COMMIT　V>之后
　　ii)

6.3.2

（a）当在<s,a,60>后插入非静态检查点，<end ckpt>可以插入在非静态检查点开始之后的任意地方，因为非静态开始检查点写入日志时，将所有已提交到缓冲区但未写到数据库中的事务数据写到磁盘。
如果错误发生在<end ckpt>后，只需要往回扫描到检查点开始的地方就可以知道哪些未完成的事务。如果错误发生在<end ckpt>前，需要往回扫描到开始非静态检查点的上一个开始非静态检查点，在这里就是扫描到整个日志的开始。
（b）当在<t,a,10>后插入非静态检查点，<end ckpt>可以插入在非静态检查点开始之后的任意地方，因为非静态开始检查点写入日志时，将所有已提交到缓冲区但未写到数据库中的事务数据写到磁盘。
如果错误发生在<end ckpt>后，只需要往回扫描到检查点开始的地方就可以知道哪些未完成的事务。如果错误发生在<end ckpt>前，需要往回扫描到开始非静态检查点的上一个开始非静态检查点，在这里就是扫描到整个日志的开始。
(c) 当在<u,b,20>后插入非静态检查点，<end ckpt>可以插入在非静态检查点开始之后的任意地方，因为非静态开始检查点写入日志时，将所有已提交到缓冲区但未写到数据库中的事务数据写到磁盘。

如果错误发生在<end ckpt>后，只需要往回扫描到检查点开始的地方就可以知道哪些未完成的事务。如果错误发生在<end ckpt>前，需要往回扫描到开始非静态检查点的上一个开始非静态检查点，在这里就是扫描到整个日志的开始。

(d) 当在<u,d,40>后插入非静态检查点，<end ckpt>可以插入在非静态检查点开始之后的任意地方，因为非静态开始检查点写入日志时，将所有已提交到缓冲区但未写到数据库中的事务数据写到磁盘。

如果错误发生在<end ckpt>后，只需要往回扫描到检查点开始的地方就可以知道哪些未完成的事务。如果错误发生在<end ckpt>前，需要往回扫描到开始非静态检查点的上一个开始非静态检查点，在这里就是扫描到整个日志的开始。

(e) 当在<t,e,50>后插入非静态检查点，<end ckpt>可以插入在非静态检查点开始之后的任意地方，因为非静态开始检查点写入日志时，将所有已提交到缓冲区但未写到数据库中的事务数据写到磁盘。

如果错误发生在<end ckpt>后，只需要往回扫描到检查点开始的地方就可以知道哪些未完成的事务。如果错误发生在<end ckpt>前，需要往回扫描到开始非静态检查点的上一个开始非静态检查点，在这里就是扫描到整个日志的开始。

6.3.4

a) <ABORT U>,<ABORT T>
b) B 写入值 20，D 写入值 40，<ABORT U>
c) B 写入值 20，D 写入值 40，<ABORT U>
d) A 写入值 10，B 写入值 20，C 写入值 30，D 写入值 40，E 写入值 50

6.4.2

a) A 写入值 10，<ABORT U>,<ABORT T>
b) A 写入值 10，B 写入值 21，C 写入值 30，D 写入值 41，<ABORT U>
c) A 写入值 10，B 写入值 21，C 写入值 30，D 写入值 41，<ABORT U>
d) A 写入值 11，B 写入值 21，C 写入值 31，D 写入值 41，E 写入值 51

6.4.4

a)
i)<COMMIT S>之后
ii)
b)
i)<COMMIT T>之后
ii)
c) i) <COMMIT T>之后
ii)
d) i) <COMMIT V>之后
ii)
e) i) <COMMIT V>之后
ii)

6.5.1

a)

<START DUMP>

<START CKPT(T1,T2)>

<T1,A,5>

<T2,C,6>

<COMMIT T2>

<T1,B,7>

<END CKPT>

DUMP COMPLETES

<END DUMP>

b)

T1 对应的日志记录不被重做

C)

(1,2,6,4)

第七章：并发控制

7.1.1

r1(A);r1(B);r1(C);w1(C);r1(D);w1(D);r1(E);w1(E)

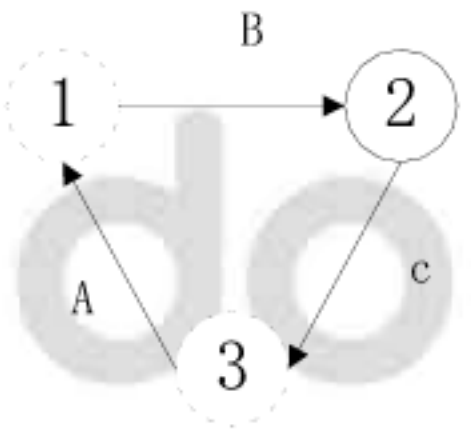
7.2.1

a)非串行调度：r1(A);w1(A);r2(A);w2(A);r1(B);w1(B);r2(B);w2(B)
b)串行调度：r1(A);w1(A);r1(B);w1(B);r2(B);w2(B);r2(A);w2(A)
c) 924
d)假设初始 A 为 2，B 为 3，若按照（T1,T2）顺序执行后 A 变为 7，B 变为 18；若按照（T2,T1）顺序执行后 A 为 7，B 为 18

7.2.5

a)

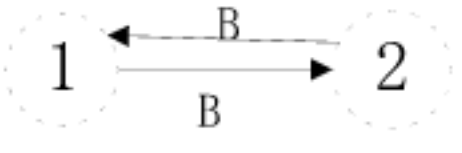
i)优先图为



Ii)该调度不是冲突可串行化的
iii)没有

b)

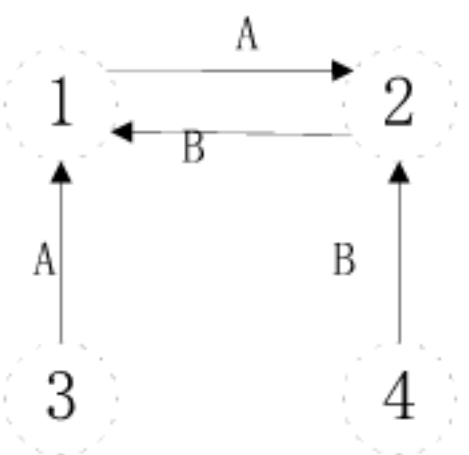
i)优先图为



ii)该调度不是冲突可串行化的
iii)有不是冲突等价的等价调度，如：r2(A);r1(A);w1(B);w2(B);r1(B);r2(B);w2(c);w1(D)

c)

i)优先图为

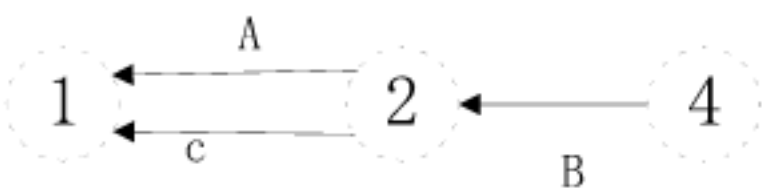


ii)该调度不是冲突可串行化的

iii)有不是冲突等价的等价调度，如：r2(A);r1(A);r1(B);r2(B);r3(A);r4(B);w1(A);w2(B)

d)

i)优先图为



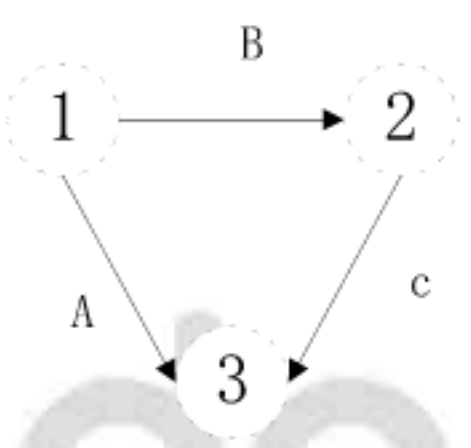
ii)是冲突可串行化的调度，等价的串行调度为：

r3(B);r2(A);r2(C);r2(B);w2(B);r1(A);w1(A);w1(C)

iii)有不是冲突等价的等价调度，如：r2(A);r1(A);r3(B);w1(A);r2(C);r2(B);w2(B);w1(C)

e)

i)优先图为



Ii)是冲突可串行化的调度，等价的串行调度为：

r1(A);w1(B);r2(B);w2(C);r3(C);w3(A)

iii)没有

7.3.1(a)

r1(A);r2(A);w1(A);r1(B);w1(B);r2(B);w2(B);w2(A)

7.3.3

在 a)中的调度，没有延迟的

在 b)中的调度，w2(B)将被延迟，r1(B)后将被允许继续

在 c)中的调度，r2(A),r3(A)将被延迟，w1(A)后将被允许继续， r4(B)将被延迟，w2(B)后将被允许继续

在 d)中的调度，r2(A)将被延迟，w1(A)后将被允许继续

在 e)中的调度，没有延迟的

7.4.1

a)

i)sl1(A);r1(A);sl2(B);r2(B);xl2(C);w2(C);u2(B);u2(C);sl3(C);r3(C);xl1(B);w1(B);u1(A);u1(B);
xl3(A);w3(A);u3(C);u3(A)

ii)事务 1 和事务 2 和事务之间会出现死锁

iii)

sl1(A);r1(A);sl2(B);r2(B);sl3(C);r3(C);xl1(B);ul1(B);ul1(A);xl2(c);w2(c);ul2(c);ul2(c);xl3(A);w3(A);ul(A);ul(C);

iv)事务 1 和事务 2 和事务 3 之间会出现死锁

v)

sl1(A);r1(A);sl2(B);r2(B);sl3(C);r3(C);xl1(B);ul1(B);ul1(A);xl2(c);w2(c);ul2(c);ul2(c);xl3(A);w3(A);ul(A);ul(C);

vi) 事务 1 和事务 2 和事务之间会出现死锁

b)

i)sl1(A);r1(A);sl2(B);r2(B);sl3(C);r3(C);sl1(B);r1(B);sl2(C);r2(C);sl3(D);r3(D);xl3(E);w3(E);u3(C);u3(D);u3(E);xl2(D);w2(D);u2(B);u2(C);u2(D);xl1(C);w1(C);u1(A);u1(B);u1(C)

ii)不会出现死锁

iii)

sl1(A);r1(A);sl2(B);r2(B);sl3(C);r3(C);sl1(B);r1(B);sl2(C);r2(C);sl3(D);r3(D);xl1(C);w(C);xl2(D);ul1(C);ul1(B);ul1(A);w2(D);ul2(D);ul2(C);ul2(B);xl3(E);w3(E);ul3(E);ul3(D);ul3(c);

iv) 不会出现死锁

v)

sl1(A);r1(A);sl2(B);r2(B);sl3(C);r3(C);sl1(B);r1(B);sl2(C);r2(C);sl3(D);r3(D);xl1(C);w(C);xl2(D);ul1(C);ul1(B);ul1(A);w2(D);ul2(D);ul2(C);ul2(B);xl3(E);w3(E);ul3(E);ul3(D);ul3(c);

vi) 不会出现死锁

7.4.2(a)

Sl1(A);r1(A);sl2(B);r2(B);xl1(C);w1(C);u1(A);u1(C);xl2(D);w2(D);u2(B);u2(D);il1(B);inc1(B);u1(B);il2(A);inc2(A);u2(A)

7.6.2

a) IS1(C);IS1(B2);S1(O5);r1(O5);IX2(C);IX2(B);IS2(C);IS2(B2);S2(O3);r2(O3);IX1(C);IX1(B2);X1(O4);w1(O4);U1(O5);U1(O4);U1(B2);U1(C);X2(O5);w2(O5);U2(O3);U2(O5);U2(B2);U2(C)

b) IS1(C);IS1(B1);S1(O1);r1(O1);IS1(B2);S1(O3);r1(O3);U1(O1);U1(O3);U1(B1);U1(C);IS2(C);IS2(B1);S2(O1);r2(O1);IX2(C);IX2(B2);X2(O4);w2(O4);X2(O5);w2(O5);U2(O4);U2(O5);U2(B2);U2(B1);U2(C)

7.7.2

a)4 种

r1(A);r1(B);r1(E);r3(B);r3(E);r3(F);

r3(B);r3(E);r3(F);r1(A);r1(B);r1(E);

r1(A); r3(B);r3(E);r3(F);r1(B);r1(E);

r3(B); r1(A);r3(E);r3(F); r1(B);r1(E);

r3(B); r3(E);r1(A); r3(F); r1(B);r1(E);

b)10 种

r2(A);r2(C);r2(B);r3(B);r3(E);r3(F);

r3(B);r3(E);r3(F);r2(A);r2(C);r2(B);

r3(B);r3(E);r2(A);r2(C);r2(B);r3(F);

r3(B);r3(E);r2(A); r3(F);r2(C);r2(B);

r3(B);r3(E);r2(A);r2(C);;r3(F);r2(B);

r2(A);r2(C); r3(B);r3(E);r3(F);r2(B);

r2(A);r2(C); r3(B);r3(E);r2(B); r3(F);

r2(A);r2(C); r3(B);r2(B); r3(E);r3(F);

r2(A); r3(B);r3(E);r3(F);r2(C);r2(B);

r2(A); r3(B);r3(E);r2(C);r2(B); r3(F);

7.9.1

a)无论什么情况，3 个事务都可以进行有效性确认。

b)fin(1)<start(2),否则事务 2 会出现验证不通过

fin(1)<start(3),否则事务 3 会出现验证不通过