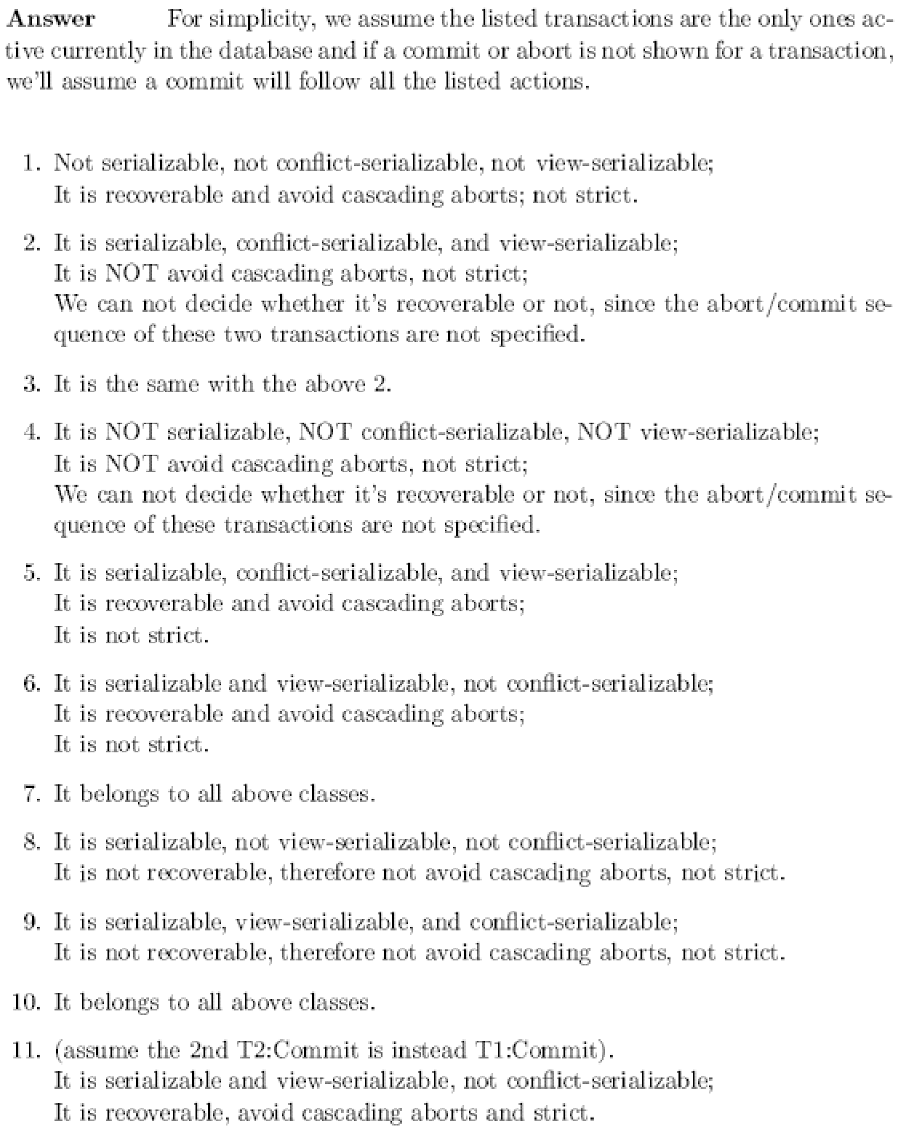
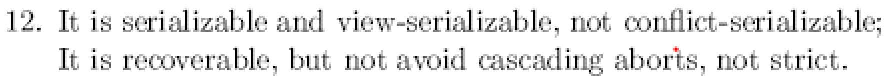
Solution 17.2



Answer For simplicity, we assume the listed transactions are the only ones active currently in the database and if a commit or abort is not shown for a transaction, we'll assume a commit will follow all the listed actions.

1. Not serializable, not conflict-serializable, not view-serializable;
   It is recoverable and avoid cascading aborts; not strict.
2. It is serializable, conflict-serializable, and view-serializable;
   It is NOT avoid cascading aborts, not strict;
   We can not decide whether it's recoverable or not, since the abort/commit sequence of these two transactions are not specified.
3. It is the same with the above 2.
4. It is NOT serializable, NOT conflict-serializable, NOT view-serializable;
   It is NOT avoid cascading aborts, not strict;
   We can not decide whether it's recoverable or not, since the abort/commit sequence of these transactions are not specified.
5. It is serializable, conflict-serializable, and view-serializable;
   It is recoverable and avoid cascading aborts;
   It is not strict.
6. It is serializable and view-serializable, not conflict-serializable;
   It is recoverable and avoid cascading aborts;
   It is not strict.
7. It belongs to all above classes.
8. It is serializable, not view-serializable, not conflict-serializable;
   It is not recoverable, therefore not avoid cascading aborts, not strict.
9. It is serializable, view-serializable, and conflict-serializable;
   It is not recoverable, therefore not avoid cascading aborts, not strict.
10. It belongs to all above classes.
11. (assume the 2nd T2:Commit is instead T1:Commit).
    It is serializable and view-serializable, not conflict-serializable;
    It is recoverable, avoid cascading aborts and strict.



12. It is serializable and view-serializable, not conflict-serializable;
    It is recoverable, but not avoid cascading aborts, not strict.

Solution 17.6

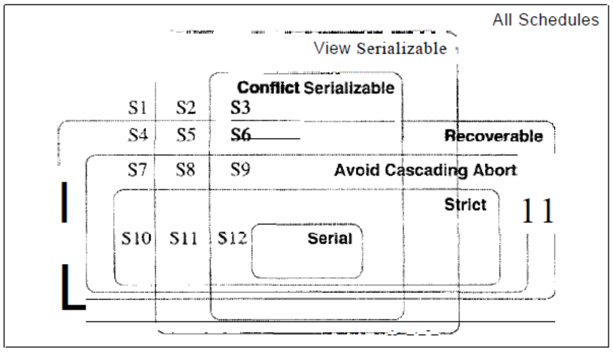


Figure 17.9 Venn Diagram for Classes of Schedules

Answer

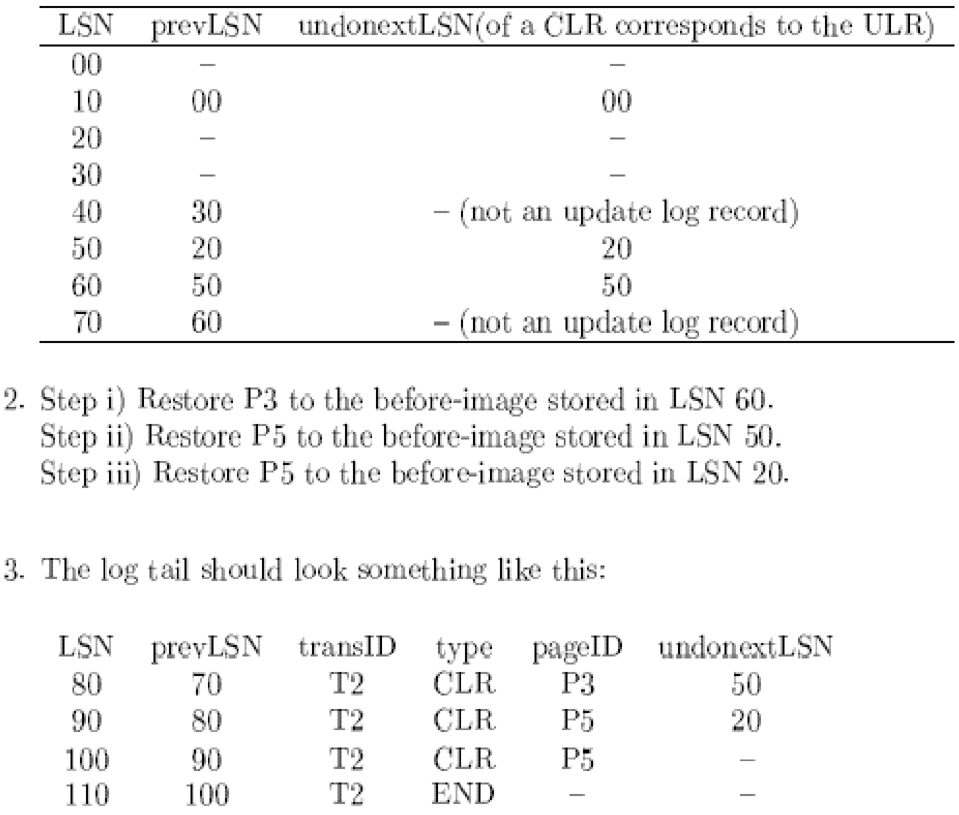
- S1
  T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Commit

- S2
  T1:R(X), T2:W(X), T1:W(X), T3:R(X), T3:Commit, T1:Commit, T2:Commit
- S3
  T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Abort
- S4
  T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y), T3:Commit, T2:Commit, T1:Commit
- S5
  T1:R(X), T2:W(X), T1:W(X), T3:R(X), T1:Commit, T2:Commit, T3:Commit
- S6
  T1:W(X), T2:R(Y), T1:R(Y), T2:R(X), T1:Commit, T2:Commit
- S7
  T1:R(X), T2:R(X), T1:W(X), T2:W(X), T1:Commit, T2:Commit
- S8
  T1:R(X), T2:W(X), T1:W(X), T2:Commit, T1:Commit
- S9
  T1:R(X), T2:W(X), T1:W(X), T2:Abort, T1:Commit
- S10
  T1:R(X), T2:R(X), T1:W(X), T1:Commit, T2:W(X), T2:Commit
- S11
  T1:R(X), T2:W(X), T2:Commit, T1:W(X), T1:Commit, T3:R(X), T3:Commit

- S12
  T1:W(X), T2:R(X), T1:W(X), T2:Abort, T1:Commit

Solution 18.4

1.



LSN	prevLSN	undonextLSN(of a CLR corresponds to the ULR)
00	–	–
10	00	00
20	–	–
30	–	–
40	30	– (not an update log record)
50	20	20
60	50	50
70	60	– (not an update log record)

2. Step i) Restore P3 to the before-image stored in LSN 60.
   Step ii) Restore P5 to the before-image stored in LSN 50.
   Step iii) Restore P5 to the before-image stored in LSN 20.

3. The log tail should look something like this:

LSN	prevLSN	transID	type	pageID	undonextLSN
80	70	T2	CLR	P3	50
90	80	T2	CLR	P5	20
100	90	T2	CLR	P5	–
110	100	T2	END	–	–