Problem 1. The answer to each question is given below.

1. $\sigma_{a<50,000}(R)$ - For this selection, the choice of accessing the sorted file is slightly superior in cost to using the clused B+ tree index simply because of the lookup cost required on the B+ tree.
2. $\sigma_{a=50,000}(R)$ - A linear hashed index should be cheapest here.
3. $\sigma_{a>50,000 \wedge a<50,010}(R)$ - A B+ tree should be the cheapest of the three.
4. $\sigma_{a\neq50,000}(R)$ - Since the selection will require a scan of the available entries, and we're starting at the beginning of the sorted index, accessing the sorted file should be slightly more cost-effective, again because of the lookup time.

Problem 2. In Pass 0, 31250 sorted runs of 320 pages each are created. For each run, we read and write 320 pages sequentially. The I/O cost per run is $2 * (10 + 5 + 1 * 320) = 670$ms. Thus, the I/O cost for Pass 0 is $31250 * 670 = 20937500$ms. For each of the cases discussed below, this cost must be added to the cost of the subsequent merging passes to get the total cost. Also, the calculations below are slightly simplified by neglecting the effect of a final read/written block that is slightly smaller than the earlier blocks.

1. For 319-way merges, only 2 more passes are needed. The first pass will produce

$$\lceil 31250/319 \rceil = 98$$

sorted runs; these can then be merged in the next pass. Every page is read and written individually, at a cost of 16ms per read or write, in each of these two passes. The cost of these merging passes is therefore $2*(2*16)*10000000 = 640000000$ms. (The formula can be read as 'number of passes times cost of read and write per page times number of pages in file'.)

2. With 256-way merges, only two additional merging passes are needed. Every page in the file is read and written in each pass, but the effect of blocking is different on reads and writes. For reading, each page is read individually at a cost of 16ms. Thus, the cost of reads (over both passes) is $2 * 16 * 10000000 = 320000000$ms. For writing, pages are written out in blocks of 64 pages. The I/O cost per block is $10 + 5 + 1 * 64 = 79$ms. The number of blocks written out per pass is $10000000/64 = 156250$, and the cost per pass is $156250 * 79 = 12343750$ms. The cost of writes over both merging passes is therefore $2 * 12343750 = 24687500$ms. The total cost of reads and writes for the two merging passes is $320000000 + 24687500 = 344687500$ms.

3. With 16-way merges, 4 additional merging passes are needed. For reading, pages are read in blocks of 16 pages, at a cost per block of $10 + 5 + 1 * 16 = 31$ms. In each pass, $10000000/16 = 625000$ blocks are read. The cost of reading over the 4 merging passes is therefore $4 * 625000 * 31 = 77500000$ms. For writing, pages are written in 64 page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 4 merging passes is $4 * 12343750 = 49375000$ms, and the total cost of the merging passes is $77500000 + 49375000 = 126875000$ms.

4. With 8-way merges, 5 merging passes are needed. For reading, pages are read in blocks of 32 pages, at a cost per block of $10 + 5 + 1 * 32 = 47$ms. In each pass, $10000000/32 = 312500$ blocks are read. The cost of reading over the 5 merging passes is therefore $5 * 312500 * 47 = 73437500$ms. For writing, pages are written in 64 page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 5 merging passes is $5 * 12343750 = 61718750$ms, and the total cost of the merging passes is $73437500 + 61718750 = 135156250$ms.

5. With 4-way merges, 8 merging passes are needed. For reading, pages are read in blocks of 64 pages, at a cost per block of $10 + 5 + 1 * 64 = 79$ms. In each pass, $10000000/64 = 156250$ blocks are read. The cost of reading over the 8 merging passes is therefore $8 * 156250 * 79 = 98750000$ms. For writing, pages are written in 64 page blocks, and the cost per pass is 12343750ms as before. The cost of writes over 8 merging passes is $8 * 12343750 = 98750000$ms, and the total cost of the merging passes is $98750000 + 98750000 = 197500000$ms.

Problem 3. 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 does NOT avoid cascading aborts, is 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 number 2 above.

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 is not serializable, not view-serializable, not conflict-serializable;
   It is not recoverable, therefore not avoid cascading aborts, not strict.

8. It is not 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.

Problem 4. The answer to each question is given below.

1. The extended figure is shown below:

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)

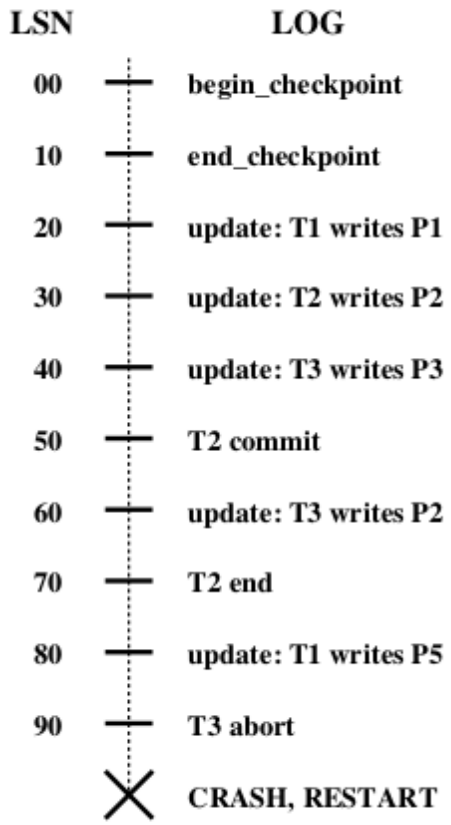

Figure 18.3 Execution with Multiple Crashes

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.

Problem 5.

1. CDE, ACD, BCD
2. R is in 3NF because B, E and A are all parts of keys.
3. R is not in BCNF because none of A, BC and ED contain a key.