**Database Systems (CS203)**

**Final Solution**

Solution 1:

1. The table violates 2nd Normal Form because there are two partial dependencies:

StudentID 🡪 StudentName and BookID 🡪 BookTitle

1. Insert anomaly: a new book cannot be added without having a student borrower associated with it. Alternatively: when inserting a new record, the title of an existing book is mistyped creating multiple copies for the same book. Also while deleting book record, the student record can be deleted.
2. Student(StudentID, StudentName)

Book(BookID, BookTitle)

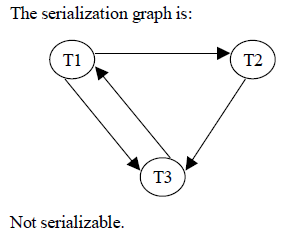
Borrowing(StudentID\*, BookID\*, Date)

Solution 2:

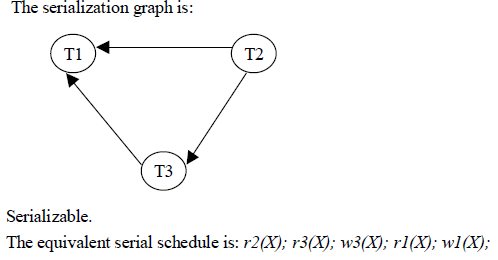
1. In S3, every transaction commits right after it writes some items. There is no write to or read from an item before the last transaction that wrote that item has committed. So S3 is **strict**.
2. In S4, T2 reads item Y from T3 but T2 commits before T3 commits. So S4 is **nonrecoverable**.
3. S5 is **not strict** because T2 writes Y before T3 commits. But S5 is **cascadeless** because there is no transaction reads items that were written by an uncommitted transaction. S5 is not serializable.

Solution 3:

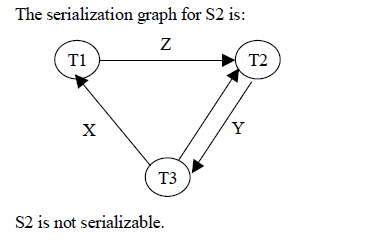
Part 1:



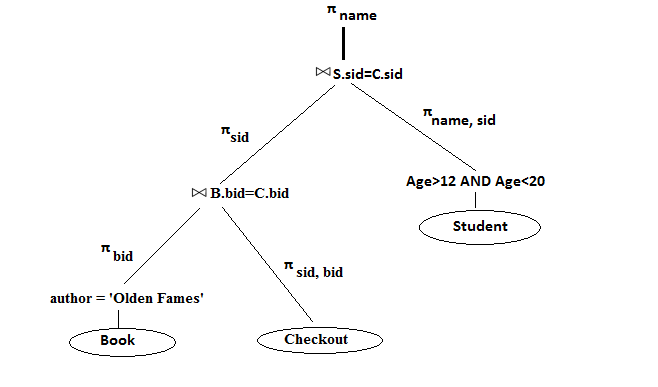
Part 2:



Part 3:



Solution 4: Optimized Query Plan



Solution 5:

1. SELECT C.pid

FROM Catalog C

WHERE EXISTS (SELECT C1.sid

FROM Catalog C1

WHERE C1.pid = C.pid AND C1.sid <> C.sid )

1. SELECT C.pid

FROM Catalog C, Suppliers S

WHERE S.sname = ‘Yosemite Sham’ AND C.sid = S.sid

AND C.cost *≥* ALL (Select C2.cost

FROM Catalog C2, Suppliers S2

WHERE S2.sname = ‘Yosemite Sham’

AND C2.sid = S2.sid)

1. SELECT C.sid

FROM Parts P, Catalog C

WHERE P.color = ‘red’ AND P.pid = C.pid

AND EXISTS ( SELECT P2.pid

FROM Parts P2, Catalog C2

WHERE P2.color = ‘green’ AND C2.sid = C.sid

AND P2.pid = C2.pid )

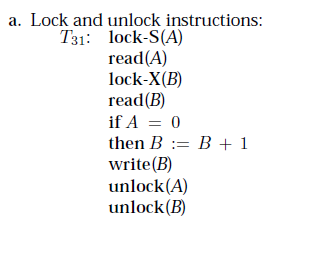
1. SELECT C1.sid, C2.sid

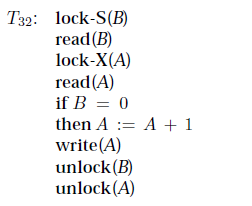
FROM Catalog C1, Catalog C2

WHERE C1.pid = C2.pid AND C1.sid <> C2.sid

AND C1.cost *>* C2.cost

Solution 6:

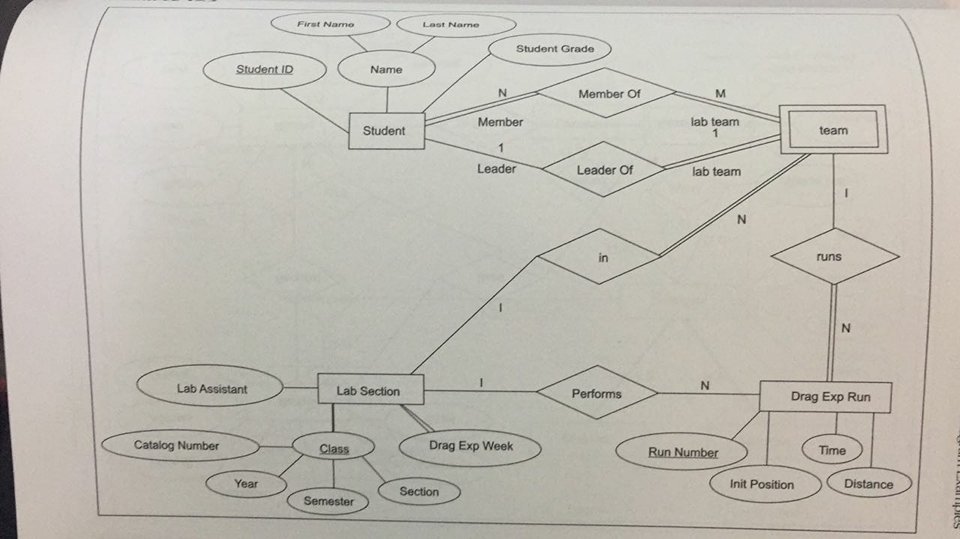




Solution 7:

1. Immediate update mode is using the (Undo/Redo) protocol for the recovery of the database state whenever a failure occur. In given scenario the system is crashing after the LSN 111.
2. The transaction T1 has reached its commit point while T2 and T3 are still in running phase. So the committed transaction has to be redone and running transaction has to be rolled back.
3. For the committed transaction T1, the system will go to first update point of the log file and roll forward to recover database from the AFIM (After image) values.
4. For the running transactions T1 and T2, the system will roll backward in order to restore to the last state of database.
5. In deferred update mode is using the (No Undo/Redo) protocol for recovery of database state whenever a failure occur. So committed transactions has to be redone

Solution 8:



Solution 9:

