1. The last check point is 2016-05-05 16:25:13, therefore nothing is going to happen before it as data was physically save to the disk. T4 and T7 transaction will be redone using the after value of the transaction. All these changes will be in ascending order, from the oldest which is T4 to the youngest, that is T7.

T1, T5, T6 and T8 operation will be undo using the before value in the transaction log. All these transaction will be in reverse order from T8 to T1 (given that T8 is the youngest).

Do nothing on T3 as changes already been made when it aborted.

2. T3: R(y)-T3 request a shared lock and gets it.

T3:W(y)-T3 request an exclusive lock and gets it.

T1:R(Z)-T1 request a shared lock and gets it

T1: W(y)-T1 request an exclusive lock on Z and gets it.

T2: R(x)-T2 request a shared lock on x, but T3 has an exclusive lock on x. This create a conflict so since T2 is older that T3 it pre-empts T3, then T3 is rescheduled using the same time stamp.

T1:W(y)- T1 request an exclusive look on Y get it. This won’t conflict as T3 already roll back.

T3: W(u)- Nothing is going to happen, T3 was rolled back.

T2: R(z)-T2 request a shared lock on z, It does not get it because T1 has locked z, therefore it wait until T1 completes and release a lock on z.

T1 commits and release all the locks.

T2 commits and release all the locks.

T3 wakeup and restart its transaction.

3.a) lost Update. The transaction change made to x, while the value x hadn’t been committed in transaction increase-marks

b) Use Two phase locking which will acquire all the locks and lock them do all necessary operations then release the locks.

4.a SELECT B.BName   
FROM BORROWER B, (SELECT DATEOUT, L.CarNo FROM Loan L, Books B WHERE B.ISBN = L.ISBN AND B.Title = “Fity Shades of Grey” FT, (SELECT DateOUt, L.CardNO FROM Loan L, Book B, WHERE B.ISBN = L.ISBN AND B.Title =”The Last Juror”)TT;

b)

Select Borrower.BName, COUNT(\*) as No\_of\_Books

FROM Borrower, Loan

WHERE Borrower.CardNo=Loan. CardNo AND Loan.DateDue>#2016/05/10#

Group by Borrower.BName

HAVING COUNT(\*)>=3;

5.a)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Plan | STEP | OPERATION | I/O OPERATION | I/O COST | RESULTING SET ROWS | TOTAL I/O COST |
| A | 1 | EMPLOYEES A1=EMPLOYEE |  | 8000 | 8000 | 8000 |
| A | 2 | Select rows in A1 with emp-gender=”F”; | 8000 | 4150 | 4150 | 12150 |
| A | 3 | Select rows A2 with emp-areacode=”041”; | 4150 | 370 | 370 | 12520 |
| A | 4 | CHECK FOR FEMALE in area Code 041 | 370 | 190 | 190 | 12710 |
| B | 1 | Go through all employees using index | 370 | 370 | 370 | 370 |
| B | 2 | Check for Females | 370 | 190 | 190 | 560 |

b) The created indexes help increase the efficiency of search by searching the relevant rows that produce the total I/O cost of 560 compared to 12710 of plan A.

6.a) Where make=”Audi”

And Car\_Sale.Staffid=Salesperson.Staffid

And Car\_Listing.ListingID=Car\_sale.ListingId

AND SaleDate>#30/04/2016#

AND SalePrice>650000

It is fast to compare on equality comparison than inequality comparison, therefore by starting with equality comparison you will get your results faster. Given that there are few Audi car that have been sold, searching the column that have Audi your data subset will be small.

b) To increase efficiency make indexing on SaleDate;

7.a) The entity as student, staff and graduation session. We have a ternary relationship between these three entity and an association class to cater for the many to many relationship.

b) Student, Staff and graduation session will have primary keys. The association class will consist of key such as studentNo<pk><fk>,staffNo<pk><fk> and graduation\_ID<pk><fk> forming a composite key.