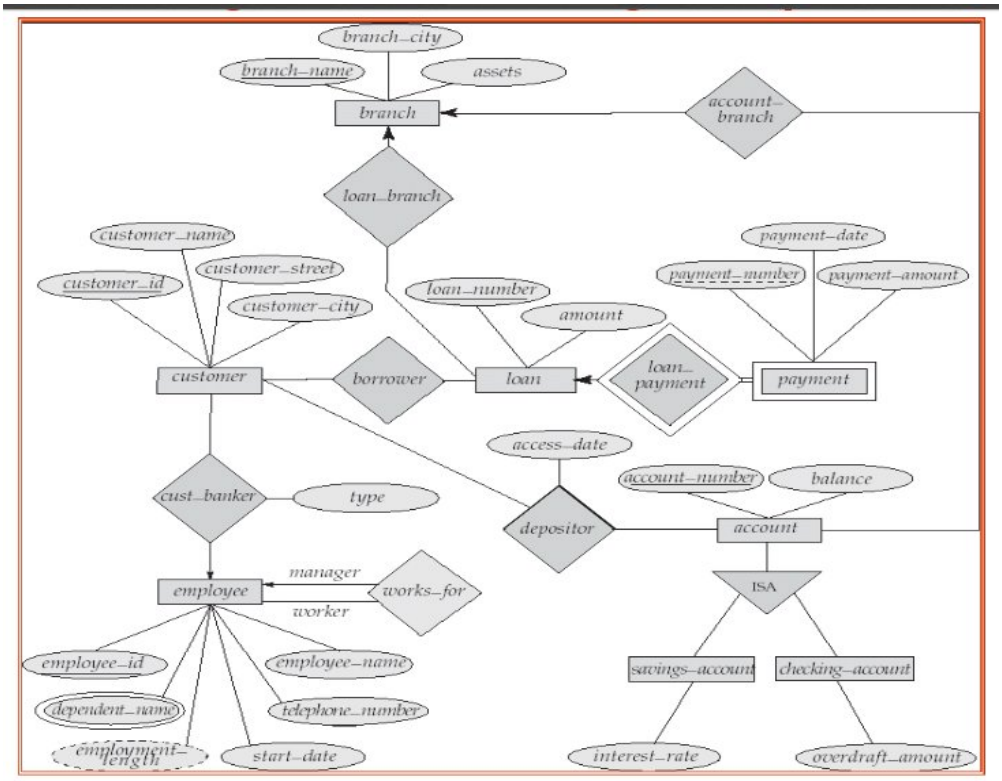
MCS-023

Qns 1: Construct an ER diagram for a Banking System. Clearly indicate the entities, relationships, cardinality and the key constraints. Also, derive the un-normalized relational database tables with the help of this diagram.

Ans:



Qns 3(a): What is non-loss decomposition in database? How it is useful in database?

Ans:

Let us show an intuitive decomposition of a relation. We need a better basis for deciding decompositions since intuition may not always be correct. We illustrate how a careless decomposition may lead to problems including loss of information.

Consider the following relation ENROL (stno, cno, date-enrolled, room-no, instructor)

Suppose we decompose the above relation into two relations enrol and enrol2 as follows:

ENROL1 (stno, cno, date-enrolled)

ENROL2 (date-enrolled, room-no, instructor)

There are problems with this decomposition but we do not wish to focus on this aspect at the moment. Let an instance of the relation ENROL be:

St no	cno	Date-enrolled	Room-no	Instructor
1123	MCS-011	20-06-2004	**1**	Navyug
1123	MCS-012	26-09-2004	**2**	Anurag Sharma
1259	MCS-011	26-09-2003	**1**	Preeti Anand
1134	MCS-015	30-10-2005	**5**	Preeti Anand
2223	MCS-016	05-02-2004	**6**	Shashi Bhushan

Figure 7: A sample relation for decomposition

Then on decomposition the relations ENROL1 and ENROL2 would be:

ENROL1

St no	Cno	Date-enrolled
1123	MCS-011	20-06-2004
1123	MCS-012	26-09-2004
1259	MCS-011	26-09-2003
1134	MCS-015	30-10-2005
2223	MCS-016	05-02-2004

ENROL2

Date-enrolled	Room-no	Instructor
20-06-2004	**1**	Navyug
26-09-2004	**2**	Anurag Sharma
26-09-2003	**1**	Preeti Anand
30-10-2005	**5**	Preeti Anand
05-02-2004	**6**	Shashi Bhushan

All the information that was in the relation ENROL appears to be still available in ENROL1 and ENROL2 but this is not so. Suppose, we wanted to retrieve the student numbers of all students taking a course from Preeti Anand, we would need to join ENROL1 and ENROL2. For joining the only common attribute is Date-enrolled. Thus, the resulting relation obtained will not be the same as that of *Figure 7*. (Please do the join and verify the resulting relation).

The join will contain a number of spurious tuples that were not in the original relation. Because of these additional tuples, we have lost the right information about which students take courses from Preeti Anand. (Yes, we have more tuples but less information because we are unable to say with certainty who is taking courses fromPreeti Anand). Such decompositions are called lossy decompositions. A nonloss or lossless decomposition is that which guarantees that the join will result in exactly the same relation as was decomposed. One might think that there might be other ways of recovering the original relation from the decomposed relations but, sadly, no other operators can recover the original relation if the join does not (why?).

We need to analyse why the decomposition is lossy. The common attribute in the above decompositions was Date-enrolled. The common attribute is the glue that gives us the ability to find the relationships between different relations by joining the relations together. If the common attribute have been the primary key of at least one of the two decomposed relations, the problem of losing information would not have existed. The problem arises because several enrolments may take place on the same date.

The dependency based decomposition scheme as discussed in the section 3.5 creates lossless decomposition.

Qns 3(b): Explain evaluation of expression process in query optimization.

Ans:

The query language processor is responsible for receiving query language statements and changing them from the English-like syntax of the query language to a form the DBMS can understand. The query language processor usually consists of two separate parts: the parser and the query optimizer.

The parser receives query language statements from application programs or command-line utilities and examines the syntax of the statements to ensure they are correct. To do this, the parser breaks a statement down into basic units of syntax and examines them to make sure each statement consists of the proper component parts. If the statements follow the syntax rules, the tokens are passed to the query optimizer.

The query optimizer examines the query language statement, and tries to choose the best and most efficient way of executing the query. To do this, the query optimizer will generate several query plans in which operations are performed in different orders, and then try to estimate which plan will execute most efficiently. When making this estimate, the query optimizer may examine factors such as: CPU time, disk time, network time, sorting methods, and scanning methods.

Qns 4 We have following relations:

Supplier(S#,sname,status,city)
Parts(P#,pname,color,weight,city)
SP(S#,P#,quantity)

Answer the following queries in SQL.

(i) Find name of supplier for city = „Delhi□.
(ii) Find suppliers whose name start with „AB□
(iii) Find all suppliers whose status is 10, 20 or 30.
(iv) Find total number of city of all suppliers.
(v) Find s# of supplier who supplies „red□ part.
(vi) Count number of supplier who supplies „red□ part.
(vii) Sort the supplier table by sname.

Ans:

(i) Select sname from Suppliers where city="delhi";
(ii) Select * from Suppliers where sname like "AB%";
(iii) Select * from Suppliers where status IN(10,20,30);
(iv) Select count(city) from Suppliers;
(v) Select Parts.color from Parts, SP where Parts.P#=SP.P# and Parts.color="red";
(vi) Select count(S#) from Supplier, Parts where Parts.color="red";
(vii) Select * from Supplier order by(sname);

Qns 4(a): Explain ACID properties of Transaction with suitable example.

Ans:

Properties of a Transaction A transaction has four basic properties. These are:

- Atomicity
- Consistency
- Isolation or Independence
- Durability or Permanence

Atomicity: It defines a transaction to be a single unit of processing. In other words either a transaction will be done completely or not at all. In the transaction example 1 & 2 please note that transaction 2 is reading and writing more than one data items, the atomicity property requires either operations on both the data item be performed or not at all.

Consistency: This property ensures that a complete transaction execution takes a database from one consistent state to another consistent state. If a transaction fails even then the database should come back to a consistent state.

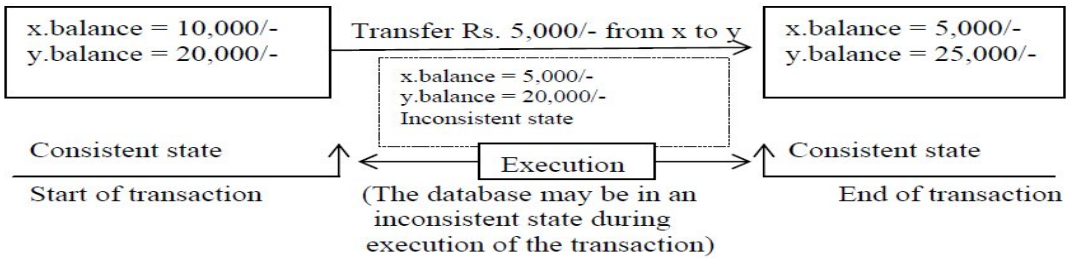


Figure 1: A Transaction execution

Isolation or Independence: The isolation property states that the updates of a transaction should not be visible till they are committed. Isolation guarantees that the progress of other transactions do not affect the outcome of this transaction. For example, if another transaction that is a withdrawal transaction which withdraws an amount of Rs. 5000 from X account is in progress, whether fails or commits, should not affect the outcome of this transaction. Only the state that has been read by the transaction last should determine the outcome of this transaction.

Durability: This property necessitates that once a transaction has committed, the changes made by it be never lost because of subsequent failure. Thus, a transaction is also a basic unit of recovery. The details of transaction-based recovery are discussed in the next unit.

Qns 4(b): Explain TWO phase locking.

Ans:

The two-phase locking protocol consists of two phases:

Phase 1: The lock acquisition phase: If a transaction T wants to read an object, it needs to obtain the S (shared) lock. If T wants to modify an object, it needs to obtain X (exclusive) lock. No conflicting locks are granted to a transaction. **New locks on items can be acquired but no lock can be released till all the locks required by the transaction are obtained.**

Phase 2: Lock Release Phase: The existing locks can be released in any order but no new lock can be acquired **after a lock has been released**. The locks are held only till they are required.

Normally the locks are obtained by the DBMS. Any legal schedule of transactions that follows 2 phase locking protocol is guaranteed to be serialisable. The two phase locking protocol has been proved for it correctness. However, the proof of this protocol is beyond the scope of this Unit. You can refer to further readings for more details on this protocol.

There are two types of 2PL:

(1) The Basic 2PL

(2) Strict 2PL

The basic 2PL allows release of lock at any time after all the locks have been acquired. For example, we can release the locks in schedule of Figure 8, after we have Read the values of Y and Z in transaction 11, even before the display of the sum. This will enhance the concurrency level.

Qns 5(b): Explain system recovery procedure with check point record.

Ans:

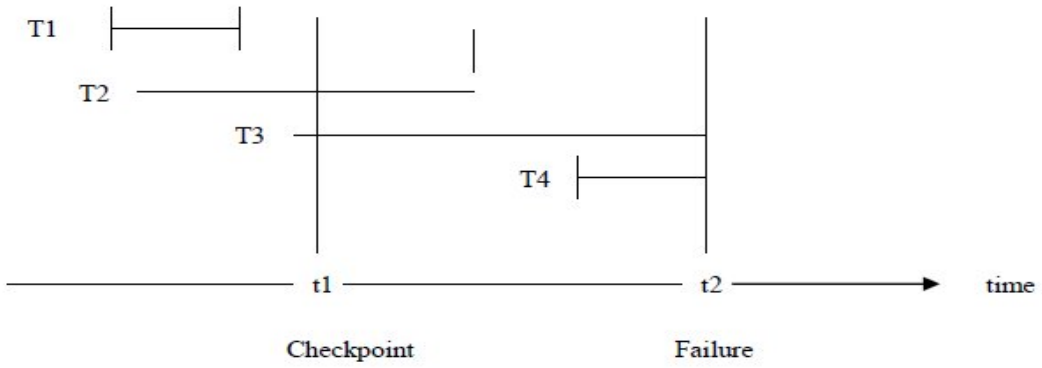


Figure 6: Checkpoint In Transaction Execution

A checkpoint is taken at time t_1 and a failure occurs at time t_2. Checkpoint transfers all the committed changes to database and all the system logs to stable storage (it is defined as the storage that would not be lost). At restart time after the failure the stable check pointed state is restored. Thus, we need to only REDO or UNDO those transactions that have completed or started after the checkpoint has been taken. The only possible disadvantages of this scheme may be that during the time of taking the checkpoint the database would not be available and some of the uncommitted values may be put in the physical database. To overcome the first problem the checkpoints should be taken at times when system load is low. To avoid the second problem some systems allow some time to the ongoing transactions to complete without restarting new transactions.

In the case of *Figure 6* the recovery from failure at t_2 will be as follows:

- The transaction T1 will not be considered for recovery as the changes made by it have already been committed and transferred to physical database at checkpoint t_1.
- The transaction T2 since it has not committed till the checkpoint t_1 but have committed before t_2, will be REDONE.
- T3 must be UNDONE as the changes made by it before checkpoint (we do not know for sure if any such changes were made prior to checkpoint) must have been communicated to the physical database. T3 must be restarted with a new name.
- T4 started after the checkpoint, and if we strictly follow the scheme in which the buffers are written back only on the checkpoint, then nothing needs to be done except restarting the transaction T4 with a new name.

The restart of a transaction requires the log to keep information of the new name of the transaction and maybe give higher priority to this newer transaction.