SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 19CS402 - DATABASE MANAGEMENT SYSTEMS

II YEAR / III SEMESTER

TRANSACTIONS

Transaction Concepts

A transaction is a logical unit of work.

- It begins with the execution of a Begin Transaction operation, and ends with the execution of a commit or rollback operation.

A sample transaction

```
BEGIN TRANSACTION :
Update acc123 { BALANCE := BALANCE – 100};
IF any error occurred Then Go to UNDO; END IF
Update acc456 { BALANCE := BALANCE + 100};
IF any error occurred THEN GOTO UNDO; END IF;
COMMIT;                          / * Successful termination */
GO TO FINISH;
UNDO :
ROLLBACK;                                  / * Unsuccessful termination */

FINISH :
RETURN;
```

The Commit operation signals successful end-of transaction.
The ROLLBACK operation signals unsuccessful end-of transaction. [something wrong]

Example

- Transaction to transfer Rs 50 from account a to account B

  Read (A)

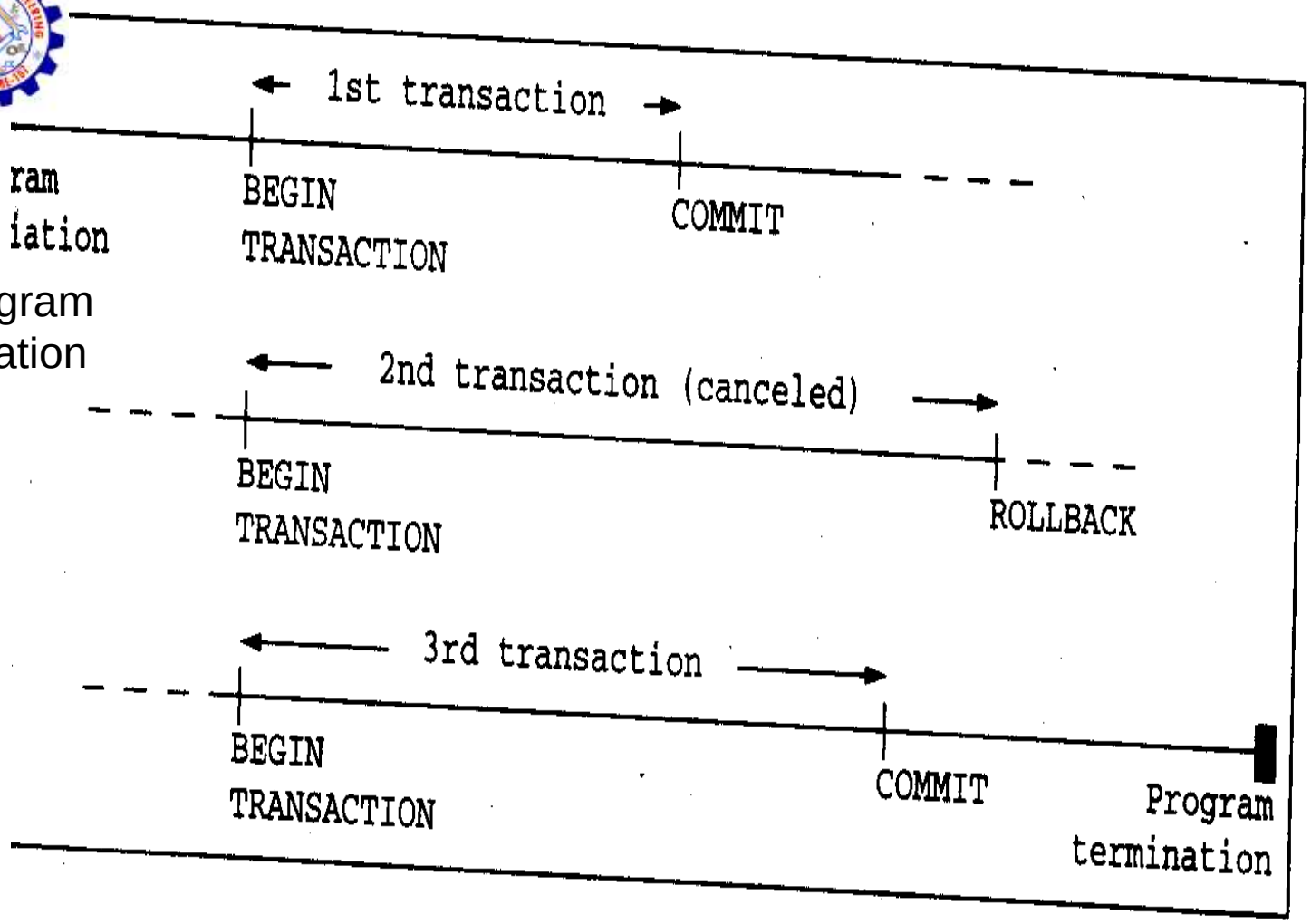
  A := A-50

  Write (A)

  Read (B)

  B:=B+50

  Write (B)



2 Program execution is a sequence of transactions

State transaction diagram

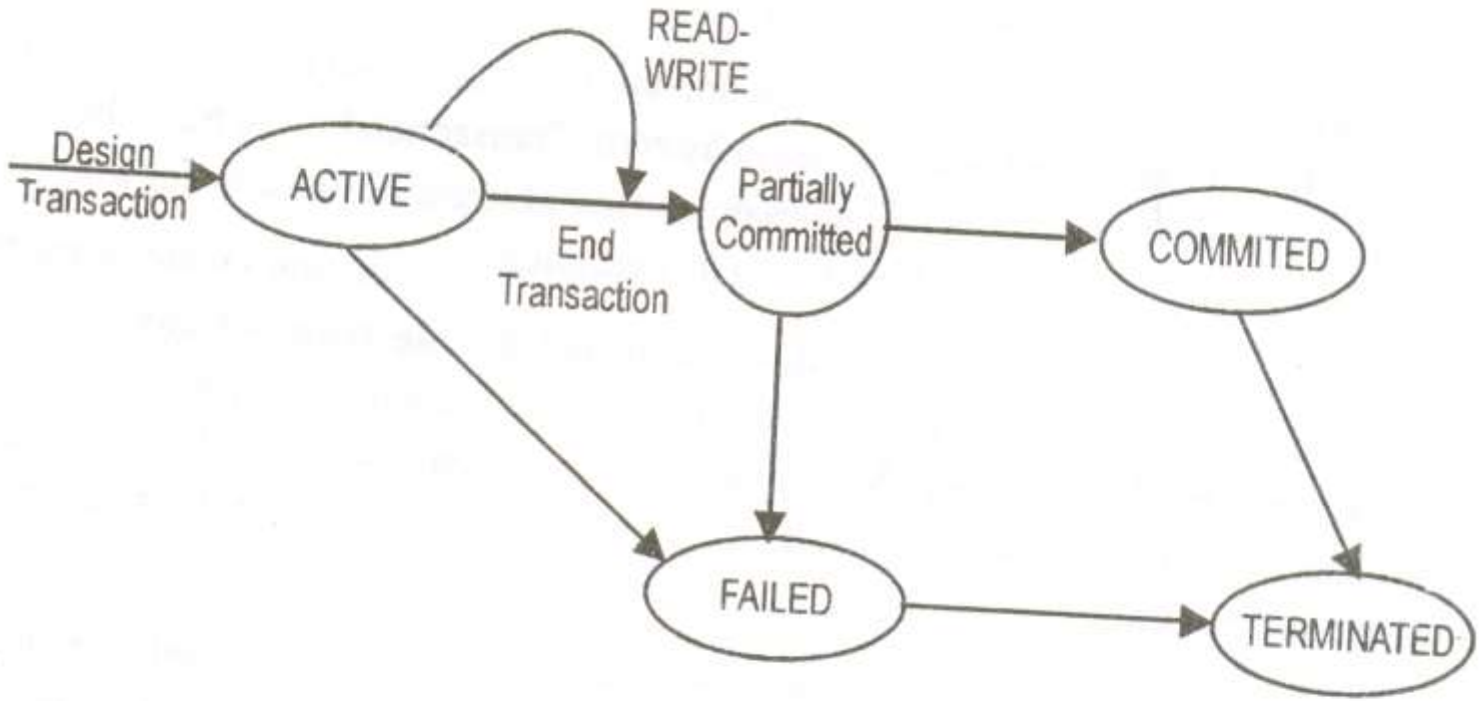


Fig. 6.2 : State transition diagram illustrating the states for transaction execution

ctive

The initial state. The transaction stays in this state while it is executing

- Partially committed
  - After the final statement has been executed
- Failed
  - After the discovery that normal execution has no longer proceed (Failed transaction)
- Aborted
  - After the transaction has been, rolled back, the database has been restored to its state prior to the start of the transaction.
- Committed
  - After successful completion.

Transaction Recovery

Transaction Recovery

- A transaction begins by executing a BEGIN TRANSACTION operation and ends by executing either a COMMIT or a ROLLBACK operation.
- Commit establishes a commit point. A commit point thus corresponds to the (successful) end of a logical unit of work, and hence to a point at which the database is supposed to be in a correct state.
- ROLL-BACK, by contrast, rolls the database back to the state it was in at BEGIN TRANSACTION, which effectively means back to the previous commit point.[or first transaction in the program]

ACID Properties

- Atomicity
  - Transactions are atomic (all or nothing)
- Correctness (Consistency)
  - Transactions transform a correct state of the database into another correct state, without necessarily preserving correctness at all intermediate points.
- Isolation
  - Transactions are isolated from one another. That is, even though in general there will be many transactions running concurrently, any given transaction's updates are concealed from all the rest, until that transaction commits.
  - Another way of saying the same thing is that, for any two distinct transactions A and B, a might see B's updates (after B has committed) or B might See A's updates (after A has committed), but not both.
- Durability
  - Once a transaction commits, its updates persist in the database, even if there is a subsequent system crash.

Types of failures

- Transaction
- System
- Media

Several reason for transaction to fail in middle of execution

- A computer failure (system crash)
  - Hardware, software or network failure.
- A transaction or system error
  - Integer overflow or division by zero, logical program error
- Local errors or execution conditions detected by the transaction
  - Notice the exception condition, such as insufficient account balance in a banking, may cause a transaction, such as find withdrawal to be cancelled.
- Concurrency control environment
  - Several transactions are in a state of deadlock.
- Disk failure
  - Disk read/write error
- Physical problems and catastrophes
  - Power failure, conditioning failure, etc

SQL Facilities
Transaction Control Language

Commit, Rollback, Savepoint

- The TCL statements give you flexibility to undo transactions or write transactions to the disk
- Transactions provide consistency in case of a system failure.

Commit

Current transaction and writes all changes permanent to the disk.

- Savepoint

  Marks a point in the current transaction

- Rollback to [savepoint n]
  - Undoing all changes
  - if n to savepoint undo the n position

Example

```
SQL> insert into emp values ( &empno, &ename, &salary,
    &dno, &comm);
SQL>/
(input some record)
SQL> select * from emp;
SQL> Commit;
SQL> Delete from emp where comm>=2500;
SQL> Select * from emp;
SQL> Rollback;
SQL> Select * from emp;
SQL> Savepoint x;
SQL> delete from emp where dno=10;
SQL> Rollback to x;
```

System Recovery

System Recovery

- The system must be prepared to recover, not only from purely local failures such as an overflow exception within an individual transaction, but also from “global” failures such as a power outage.
- Local failure, by definition, affects only the transaction in which the failure has actually occurred.
- A global failure, by contrast, affects all of the transactions in progress at the time of the failure and hence has significant system wide implications

Two categories

- System failures
  - Which affect all transactions currently in progress but do not physically damage the database. A system failure is sometimes called a soft crash.
- Media failures
  - Which do cause damage to the database or some portion thereof, and at least those transactions currently using that portion. A media failure is sometimes called a hard crash.

- A system failure has occurred at time tf
- The most recent checkpoint prior to time tf was taken at time tc.

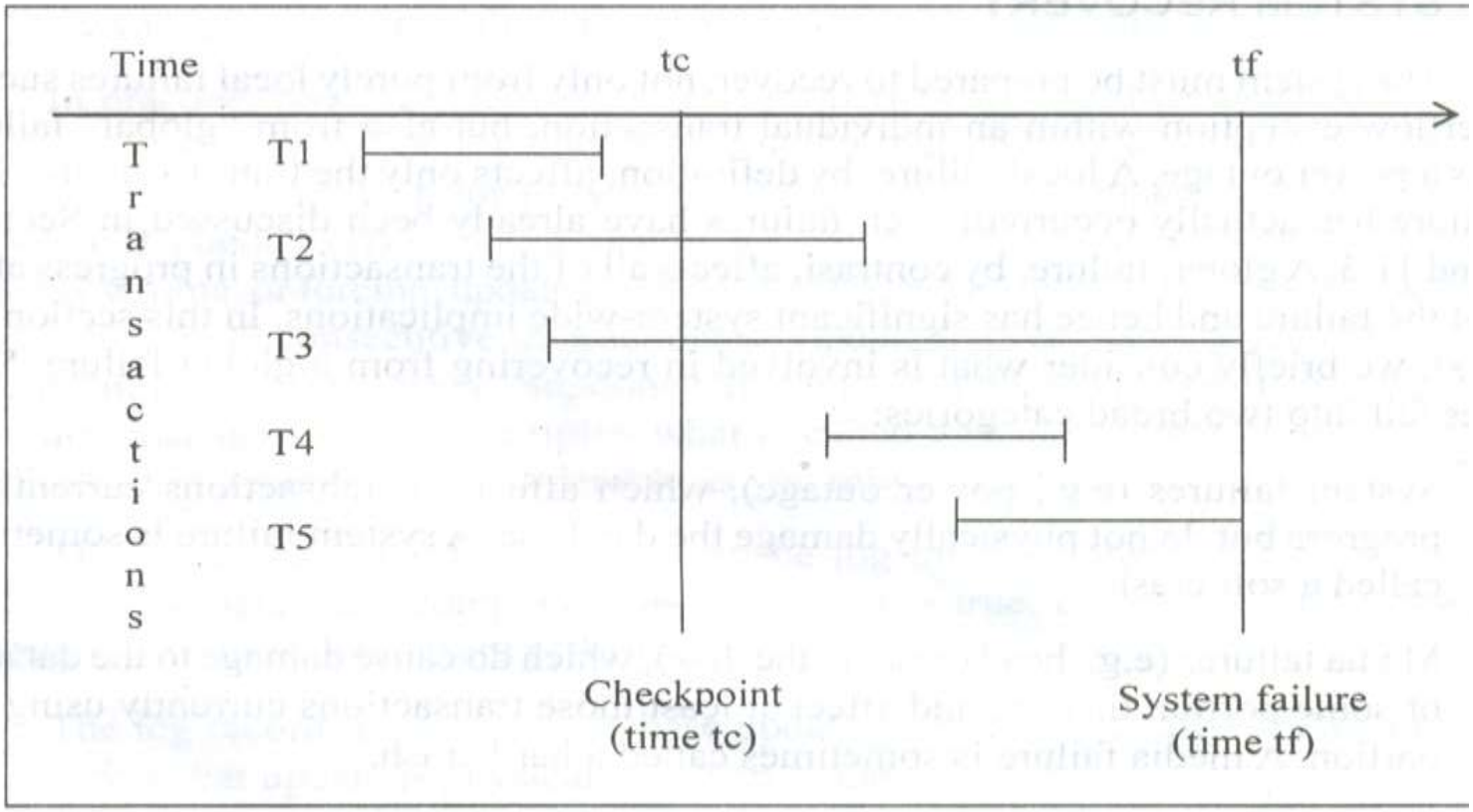


Fig. 11.5 Five transaction categories

- Transaction of type T1 completed (successfully) prior to time tc.
- Transactions of type T2 started prior to time tc and completed (successfully) after time tc and before time tf.
- Transactions of type T3 also started prior to time tc but did not complete by time tf.
- Transactions of type T4 started after time tc and completed (successfully) before time tf.
- Finally, transactions of type T5 also started after time tc but did not complete by time tf.

It should be clear that when the system is restarted, transactions of types T3 and T5 must be undone, and transactions of types T2 and T4 must be redone. Note however, that transactions of type T1 do not enter into the restart process at all because their updates were forced to the database at time tc as part of the did point process. Note too that transactions that completed unsuccessfully (ie., with a rollback) before time tf also do not enter into the restart process at all.

1. Start with two lists of transactions, the UNDO list and the REDO list.
2. Set the UNDO list equal to the list of all transactions given in the most checkpoint record and the REDO list to empty.
3. Search forward through the log, starting from the checkpoint record.
4. If a BEGIN TRANSACTION log record is found for transaction T, add T the UNDO list.
5. If a COMMIT log record is found for transaction T, move T from the list to the REDO list.
6. When the end of the log is reached, the UNDO and REDO lists identify, lively, transactions of types T3 and T5 and transactions of types T2 and T4.

- Restoring the database to a correct state by redoing work is sometimes called forward recovery.
- Similarly, restoring it to a correct state by undoing work is sometimes called backward recovery.