

# *Failures In Distributed Database System & Distributed Database Recovery*

RAMNA SATTAR



# Failures In Distributed Database System

## Failures In Distributed Database System:

A database management system is susceptible to a number of failures. In a distributed database system, failures can be broadly categorized into followings:

1. **Soft failures**
2. **Hard failures**
3. **Network failures.**



# Failures In Distributed Database System

## Soft Failure

Soft failure is the type of failure that causes the loss in volatile memory of the computer and not in the persistent storage. Here, the information stored in the non-persistent storage like main memory, buffers, caches or registers, is lost. They are also known as system crash. The various types of soft failures are as follows:

- **Operating system failure.**
- **Main memory crash.**
- **Transaction failure or abortion.**
- **System generated error like integer overflow or divide-by-zero error.**
- **Failure of supporting software.**
- **Power failure**



# Failures In Distributed Database System

## Hard Failure

A hard failure is the type of failure that causes loss of data in the persistent or non-volatile storage like disk. Disk failure may cause corruption of data in some disk blocks or failure of the total disk. The causes of a hard failure are:

- **Power failure.**
- **Faults in media.**
- **Read-write malfunction.**
- **Corruption of information on the disk.**
- **Read/write head crash of disk.**

Recovery from disk failures can be short, if there is a new, formatted, and ready-to-use disk on reserve. Otherwise, duration includes the time it takes to get a purchase order, buy the disk, and prepare it.



# Failures In Distributed Database System

## Network Failure:

Network failures are prevalent in distributed or network databases. These comprises of the errors induced in the database system due to the distributed nature of the data and transferring data over the network. The causes of network failure are as follows –

- **Communication link failure.**
- **Network congestion.**
- **Information corruption during transfer.**
- **Site failures.**
- **Network partitioning.**



# Distributed Database Recovery

## Distributed Database Recovery:

In order to recuperate from database failure, database management systems resort to a number of recovery management techniques.

The typical strategies for database recovery are :

- In case of soft failures that result in inconsistency of database, recovery strategy includes transaction undo or rollback. However, sometimes, transaction redo may also be adopted to recover to a consistent state of the transaction.
- In case of hard failures resulting in extensive damage to database, recovery strategies encompass restoring a past copy of the database from archival backup. A more current state of the database is obtained through redoing operations of committed transactions from transaction log.



# Distributed Database Recovery

## Recovery from Disk Failure

A disk failure or hard crash causes a total database loss. To recover from this hard crash, a new disk is prepared, then the operating system is restored, and finally the database is recovered using the database backup and transaction log. The recovery method is same for both immediate and deferred update modes.

The recovery manager takes the following actions:

- The transactions in the commit list and before-commit list are redone and written onto the commit list in the transaction log.
- The transactions in the active list and failed list are undone and written onto the abort list in the transaction log.



# Distributed Database Recovery

## Recovery from Power Failure

Power failure causes loss of information in the non-persistent memory. When power is restored, the operating system and the database management system restart. Recovery manager initiates recovery from the transaction logs.

In case of immediate update mode, the recovery manager takes the following actions:

- Transactions which are in active list and failed list are undone and written on the abort list.
- Transactions which are in before-commit list are redone.
- No action is taken for transactions in commit or abort lists.

In case of deferred update mode, the recovery manager takes the following actions :

- Transactions which are in the active list and failed list are written onto the abort list. No undo operations are required since the changes have not been written to the disk yet.
- Transactions which are in before-commit list are redone.
- No action is taken for transactions in commit or abort lists.





# Distributed Database Recovery

## Checkpointing:

Checkpoint is a point of time at which a record is written onto the database from the buffers. As a consequence, in case of a system crash, the recovery manager does not have to redo the transactions that have been committed before checkpoint. Periodical checkpointing shortens the recovery process.

The two types of checkpointing techniques are :

- **Consistent checkpointing**
- **Fuzzy checkpointing**



# Distributed Database Recovery

## Consistent Checkpointing:

Consistent checkpointing creates a consistent image of the database at checkpoint. During recovery, only those transactions which are on the right side of the last checkpoint are undone or redone. The transactions to the left side of the last consistent checkpoint are already committed and needn't be processed again. The actions taken for checkpointing are :

- The active transactions are suspended temporarily.
- All changes in main-memory buffers are written onto the disk.
- A “checkpoint” record is written in the transaction log.
- The transaction log is written to the disk.
- The suspended transactions are resumed.

If in step 4, the transaction log is archived as well, then this checkpointing aids in recovery from disk failures and power failures, otherwise it aids recovery from only power failures.



# Distributed Database Recovery

## Fuzzy Checkpointing:

In fuzzy checkpointing, at the time of checkpoint, all the active transactions are written in the log. In case of power failure, the recovery manager processes only those transactions that were active during checkpoint and later. The transactions that have been committed before checkpoint are written to the disk and hence need not be redone.



# Distributed Database Recovery

## Transaction Recovery Using UNDO / REDO:

Transaction recovery is done to eliminate the adverse effects of faulty transactions rather than to recover from a failure. Faulty transactions include all transactions that have changed the database into undesired state and the transactions that have used values written by the faulty transactions.

Transaction recovery in these cases is a two-step process :

- UNDO all faulty transactions and transactions that may be affected by the faulty transactions.
- REDO all transactions that are not faulty but have been undone due to the faulty transactions.



# Distributed Database Recovery

## Steps for the UNDO operation are :

- If the faulty transaction has done INSERT, the recovery manager deletes the data item(s) inserted.
- If the faulty transaction has done DELETE, the recovery manager inserts the deleted data item(s) from the log.
- If the faulty transaction has done UPDATE, the recovery manager eliminates the value by writing the before-update value from the log.

## Steps for the REDO operation are :

- If the transaction has done INSERT, the recovery manager generates an insert from the log.
- If the transaction has done DELETE, the recovery manager generates a delete from the log.
- If the transaction has done UPDATE, the recovery manager generates an update from the log.



*THANK YOU*

ANY QUERY???

