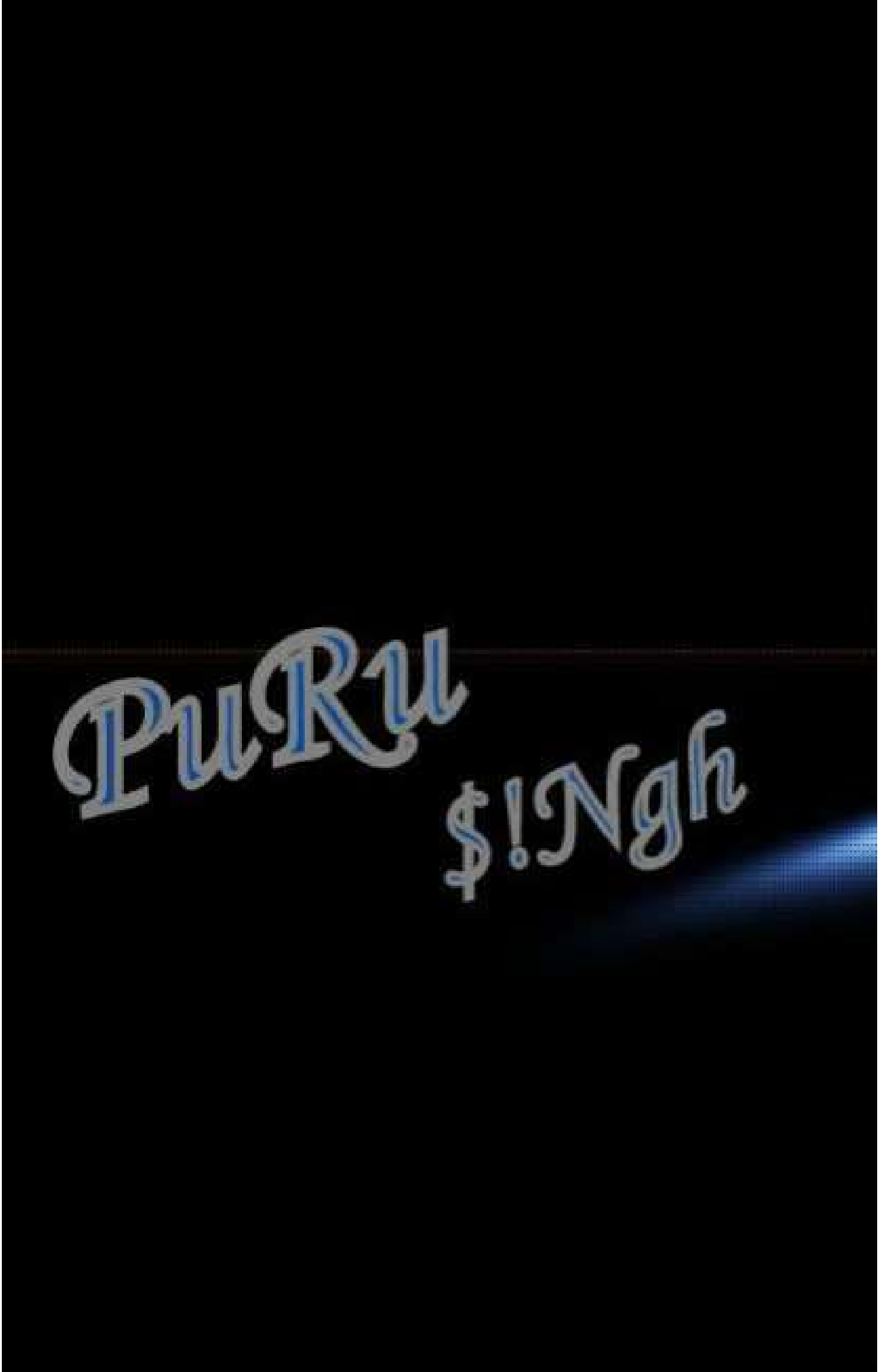
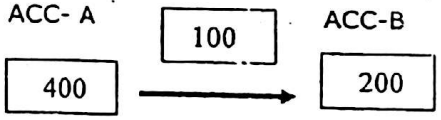

PuRu
$!Ngh

Unit – I (Transaction Management)

- Such problems are arise when Transaction management are in working position
  - The system might crash in the middle of executing some program, thereby leaving the database in an unknown state.
  - When two program executing at the same time ("concurrently") might interfere with each other and hence produce incorrect result, either in database or in the outside world.
- So in this situation Transaction management need some process to protect the data so there are Recovery & Concurrency are used to protect data from losses.
- Both are used to Data protection in Transaction Management.
- They are protecting data in the database against Loss and Damage.
- **Transaction**
  - It is the logical unit of work..
  - Its begins with the execution of the BEGINE TRANCATION operation, and end with the execution of the COMMIT or ROLLBACK operation.
  - **Example :-**
  - ACC- A



Final-300

  - It may be possible database is in incorrect state between those two update.
  - It doesn't reflect a valid state of affairs in real world.
  - Its not possible in real world but suppose we assume it will happen.
  - Transaction not happen or involve just single or one database operation.
  - Its involves sequence of operation.
  - That sequence is to transform a correct state of the database into another state , without
  - In such case it happen to update one ACC & execute & the other not because that would leave the database in an incorrect state.
  - It will happen that a system crash might occur between the two updater or an arithmetic overflow might occur.
  - A system that support transaction management does provides the next best thing to such guarantee.

- It guarantee that if the transactio execute some update ant then a failure occur before truncation reaches its planned termination than those update will undone.
- In that situation transaction either execute in its entirely or is totally cancelled.
- The system component that provides this process is known as **Transaction Manger &** also known as **Transaction Processing Monitor or TP Monitor .**
    - Commit
    - Rollback
- Are two key to do transaction management.

1) **Commit:-**

- This operation signals successful end- of – transaction.
- Its tells transaction manager that logical unit work is successfully completed.
- The database should now be in correct state.
- All updates made by TP is successfully completed.

2) **Rollback:-**

- This operation signals unsuccessful end-of- transaction.
- Truncation manager that something has go wrong.
- The database might be in an incorrect state.
- Logical unit of work so far must be rollback (undone).

➢ Some important point related to transaction management.

➢ **Implicit Rollback:**

- Its includes explicit tests for error and issues an explicit Rollback is the error is detected.
- But in real word the transaction always includes explicit test for all possible errors & the system issues a Implicit rollback.

➢ **Message Handling:-**

- A typical transaction will not only update the database it will also send some kind of message back to the end user including what has happen.

➢ **Recovery Log:-**

- If the system is crash & failure then how the transaction is undone and update.
- For that System maintain LOG or JOUNRAL on tape or disk in that they store the detail of all update.

[QUE:- Explain Common Design Errors in DSS.]

Common Designing Errors:-

1) Duplicated Rows :-

- In some situation the data are not in unique form and data does not have unique identifier.
- At that time the data create in duplicated form.
- And the situation arise at the time of physical designing of database not based on logical design.

2) Denormalization and Related Practice :-

- In some situation limited joins and reduce I/O , designer often pre-join tables , introduce derived column of various kind.
- But its not done at physical level and its only detected at the logical level.

3) Star schemas [Dimension Schema] :-

- There are lots of short cut techniques are used to retrieve the data but
- In such techniques are not helpful for performance and flexibility so this type of techniques called Star schema.

4) NULLS:-

- Designer often attempt to save space by putting nulls in column.

5) Designing of Summarized tables:

- user cant understood the path now the aggregate function is worked and its to
- confusing at logical level to create a summary tables without ignoring redundancy & difficulties to maintain of summary data with consistency.

6) Multiple navigation path :-

- multiple navigation path are used to retrieve meaningful data.

Statement Atomicity:-

- The system must guarantee that individual statements & individual execution that is atomicity.
- This consideration for only relational System.
- Where statement are set of level & typically operate many tuples at a time.

➢ **Program Execution is a Sequence of Truncation:-**

- A single program execution will consist of sequence of several transaction running one after another.

➢ **No Nested Transaction:-**

- Application program can execute a begin transaction statement only when it has no transaction processing.

➢ **Correctness:-**

- The database always at least correct state means transaction should be in complete with correct state.

➢ **Multiple Assignment :-**

- Multiple form of assignment , which allows any number of individual assignment [update] to be perform simultaneously.

[Que:- What is Recovery? Explain types of Recovery.]

Recovery:-

➢ it means recover the database up till the correct state after the failure occurs

➢ There are two types of Recovery.

1) Transaction Recovery

2) System Recovery

Transaction Recovery

- Recovery means recover the database up till the correct state after the failure occurs.
- Commit establish a COMMIT point [IN older system is known SYNCH POINT]
- A commit point are attach with successful end of logical Unit of Work & its Represent database is in correct state.
- Rollback are process on database and start with Either BEGIN TRANSCATION or Previous Commit Point.
- Condition is that the Pervious Commit Point is still Accurate.
- When Commit point is establish

1) All database Update made by the executing program since the previous commit point are Committed.
    - They are made perment , in the the sense that they are guaranteed to be data are in Correct state.

2) All database positioning is lost & all tuple locks are released.
   - Means at program executing time program have addressability to certain tuples in database.

✓ Transaction is not only a unit of work but also a unit of recovery work.

✓ Few implementation issues are discussed at Recovery stage.

✓ It should be clear that implementation will be simplest if both of the following are a true

1) Database updates ate kept in Buffer in Main Memory & physically written to disk until transaction commits.
   - If Transaction terminates unsuccessfully their will be no need to undo.
2) Database updates are physically written to disk as a part of the process of transaction with commit request.
   - If system crashes, we can be sure there will be no need to do again (REDO) any updates.

► **Write ahead log rules: -** means that the log must be physically written before commit processing can complete.

[Que:- Short note on ACID Properties]

► The **ACID properties** for Transaction Process.

⇒ **A- Atomicity** : Transaction are atomic means transaction are followed in sequence.

⇒ **C- Correctness:** Truncation transform a correct state to another correct state without preserving necessary intermediate point.

⇒ **I- Isolation:** Transaction are isolated from one to another.
   - In general case many transaction running concurrently so in that situation Transaction updates are Hidden from all the other until the transaction committed.
   - Take two transaction A & B
   - A might see B's update until B has committed.
   - B might see A's update until A has committed but not both.

⇒ **D-Durability :** once a transaction Commits , its Updates are stored in database.
   - Till the system Crash.

[Que:- What is Failure? How many Types of Failure occur? Explain system recovery.(oct/nov-2007)]

SYSTEM RECOVERY

⇒ It must be prepared to recover , not only local failure but also global failure.

⇒ **Local Failure:-** means Overflow exception within individual transaction.

- Its affects only the single transaction in which the failure has actually occure.

⇒ **Global Failure:-** means power outage (Failure).

- Affects all of the transaction in progress at the time of the failure.
- Global failure falls into two broad categaries.

1) System failure (Power failure)

- Which affects all transaction correcting in progress but do not damage physically the database.
- A system failure is sometimes called **Soft crash.**
- In system failure the contents of main memory are lost.
- So in that case when system is restarted the transaction undone or Restart.
- In some cases the transaction are redo (Do again) at restart time but did not manage to get update transferred from the buffer to physical database.
- **The main question is that how does system known at restart time which transaction is Undone & which is Redo?**
- When number of record have been written to the LOG- the system automatically makes a checkpoint.
- In involving checkpoint
  - Forcing the contents of main memory buffer from physical database.
  - Forcing a special CHECKPOINT RECORD from physical database
- **Checkpoint Record:-** contains list of all transaction that were in progress at a time when checkpoint was made.
- **Forward Recovery:-** restoring the database to a correct state by redoing works is sometimes called Forward recovery.
- **Backward Recovery:-** restoring the database to a correct state by undoing works is sometimes called backward recovery.

2) Media failure (Head crash on the Disk)

- In media failure damage to the database or some portion of the database.
- And affect at least those transaction which are currently using this portion.
- A media failure sometimes called **Hard crash.**

[Que:- Short note on TWO PHASH COMMIT

Or ***Explain Two PHASE Commit Protocol]***

TWO-PHASE COMMIT:-

- ✓ To understand commit/Rollback concept first understand Two-phase Commit.
- ✓ Its Important whenever a given transaction can Interact with several Implementation **"Resource Manager "**
- ✓ Each Resource Manager managing its own set of recoverable resources and maintaining its own recovery log.
- ✓ **For example**:- Take two transaction running on IBM framework that updates both IMS database & DB2 database.
- ✓ If the truncation completed successfully then its updates store with Commit.
- ✓ Or if it fails then all updates must be roll back.
- ✓ Means it must not be possible for the IMS updates are committed & DB2 updates are Rollback or vice versa.
- ✓ Commit and Rollback are handled by system component called **"Coordinator"**.
- ✓ Coordinators Task is to take guarantee from resource manager to commit & rollback the updates which they made in agreement even if the system is fails middle of process.
- ✓ Two phase Commit Protocol that enables the coordinator to provide such guarantee.
- ✓ On receiving commit request the coordinator goes through the following Two Phase Process.
- ✓ This are the Two phase:→
  - ➢ **Prepare :-**
    - It instruct all resources manager to get ready to **"go either way"** on the transaction.
    - Means each participant in process & each resource manager is include they forces all log records on local resources which are used by transaction to its own physical log.
    - If the force write successfully resource manager replays " OK " to coordinator otherwise its replays "Not ok"
  - ➢ **Commit :-**
    - Whenever coordinator received replies from all participants.
    - It forces records to its own physical log, & record its decision regarding transaction.
    - It replies were "OK" that take decision is "Commit "
    - If any reply is "NOT OK" then the decision is "ROLL BACK".
- ✓ If the system fails at some point during the following process then restart procedure will look for the decision records in the coordinators log.

- ✓ If it finds it then two phase commit process pick up where it left out.
- ✓ Or if it does not find it then assume that the decision was "ROLL BACK " , & again process can complete appropriately.

LOCKING:→

- ✓ "Locking means lock the transaction to prevent them from changing."
- ✓ There are two type of lock:
  - ➢ Exclusive locks (X locks)→ Write locks
  - ➢ Shared locks (S locks) → Read locks
- ✓ If a transaction T1 has obtained a shared mode lock [depended by S] then T1 can read but can not write.
  - ➢ **EXCLUSIVE LOCKS →**
    - If a transaction T1 has obtain an exclusive mode lock then T1 can both read & Write.
  - ➢ **SHARED LOCKS →**
    - Shared mode is compatible with shared mode but with exclusive mode
  - ➢ At any time , several shared mode locks can be help simultaneously on a particular data item.
  - ➢ A subsequent exclusive mode lock request has to wait until the currently had shared mode locks are released.

[Que :- Short note on INTENT LOCKING]

Intent Locking:-

- ➢ By using intent locking , we can apply more locks in a tables on particular column.
- ➢ In Intent locking , we can apply 5 different lock which are
  1) Shared lock (s)
  1) Exclusive lock (x)
  1) Intent shared (IS)
  1) Intent Exclusive (IX)
  1) Shared Intent Exclusive (SIX)

- ➢ **Intent shared (IS)** :- If transaction T1 set shared locks on individual tuples in Relation in order to stability of those tuples while they are being processed.
- ➢ **Intent Exclusive (IX)**:- same as IS , but in this lock Transaction T1 also update individual tuples in relation.
- ➢ **Shared intent Exclusive (SIX)** :- Its is combination of shared & intent Exclusive means S& IX.

[Que:- Short note on DEADLOCKS]

DEAD LOCKS:→

- Dead locks is a situation in which two or more transaction are in a simultaneous wait state.
- Each of them waiting for one of the other to release a lock before it can processed.
- If a deadlock occurs it is desirable that the system detect it and break it.
- Breaking the dead lock involves choosing one of the dead locked transaction .
- Example:-
  - One of the transaction in the cycle in the graph as the victim and rolling it back. There by releasing its locks and so allowing some other transaction to proceed.

[Que:-what is concurrency? What problems are associated with concurrency ? Explain any one]

Concurrency (What is Concurrency?)

The term concurrency refers to the fact that many truncations to access the same database of the same time.

Concurrent truncations do not interfere with each other.

Explain three concurrency problems with example.

The three problems are

1) The last update problem
2) The uncommitted dependency problem
3) The inconsistent analysis problem

3) **The Last Update Problem:-**

Transaction A	Time	Transaction B
	↓	==
Retrieve T	T1	==
==	T2	Retrieve T
Update T	T3	==
==	↓	==
==	T4	Update T
	↓	

[Figure 1]

Figure shows that transactions A retrieve some tuple t at time t1;

- truncation B retrieves that same tuple t at time t2;
- truncation A update the tuple at time t3;
- truncation B update the same tuple at time t4;

Truncation as update is lost at time t4, because truncation B overwrites it without even looking at it

3) The Uncommitted dependency problem:-

- The uncommitted dependency problem arise if one truncation is allowed to retrieves or wares update.–a tuple that has been update by another truncation but not get committed by that other truncation.

Transaction A	Time	Transaction B
==	↓	==
==	T1	Update T
==	↓	—
Retrieve T	T2	==
==	↓	==
==	T3	Roll Back
	↓	

- Above figure shows that transaction A see an uncommitted update at time t2 , that update is then undone at time t3,Transcation A is therefore operation on a False Assumption.
- The assumption that tuple T1 has the value seen at time T2, Infact it has whatever value it had prior to time T1.As a result transaction A might well Produce incorrect output.

Transaction A	Time	Transaction B
==	↓	==
==	T1	Update T
==	↓	—
Update T	T2	==
==	↓	==
==	T3	Roll Back
	↓	

[Truncation A Update an Uncommitted change at time at time T2,and losses that update at time T3]

3) The Inconsistent Analysis Problem:-

- Figure 1 shows two transaction A&B operating on account [ACC] tuple.
- Truncation A is summing account balance truncation B is Transferring an account 10 from account 3 to account 1.
- The result produce by A-110 is obviously incorrect if A were to go on to write that result back into the database.
- It would actually leave the database in an inconsistent state.
- We say that a has seen an inconsistency state of the database & has therefore performed an inconsistency analysis.

ACC1	ACC2	ACC3
40	50	30

Transaction A	Time	Transaction B
==	↓	==
Retrieve ACC1:	T1	==
SUM=40	↓	—
Retrieve ACC2:	T2	==
SUM=90	↓	==
==	T3	Retrieve ACC3
—	↓	—
—	T4	Update ACC3
—	↓	30→20
—	T5	Retrieve ACC1
—	↓	—
—	T6	Update ACC1
—	↓	40→50
—	↓	—
—	T7	Commit

—	↓	—
Retrieve ACC3	T8	—
Sum=110,Not 120	↓	—

SECURITY

[Que:- why security is necessary in the system? Which two approaches are used? Explain]

- Security refers to the protection of data against unauthorized discloser, alternation or destruction Integrity refers to the accuracy or validity of that data.
- Security means protecting data against unauthorized users.
- Integrity means protecting it against authorized user.
- Security means making sure user are allowed to do the things they are trying to do.
- There are some security problems the following among them :
    - Legal , social & Ethical aspects.
    - Physical controls
        - Example:- is the computer or terminal room locked or otherwise guarded?
    - Policy control
    - Operational problem
        - Example:-if a password scheme is used , how are the password themselves kept secret?
    - Hardware control
    - Operating system support
- Now, Morden DBMS typically support either or both of two broad approaches to data security. The approaches are known as Discretionary & Mandatory control.
- In the case of discretionary control, a given user with typically have different access right on difference object (for example, User U1 might be able to see A but not B , while user U2 might be able to see B but Not A)
- In the case of mandatory control , by contrast , each data object is labelled with a certain classification level and each user is given a certain clearance level (for example if user U1 can see A but not B, then classification of B must higher than that of A and so no user U2 see B but not A)

4+8

This is Two approaches with are used with Security:-

1)Discretionary Access Control

2)Mandatory Access Control

Discretionary Access Control:-

- Most DBMSs support either discretionary control or mandatory control or both. In fact it would be more accurate to say that most system support discretionary control and some system support mandatory control as well.
- The point that in general authority have four components:

```
Authority SA3
   GRANT RETRIEVE (S#,SNAME,CITY),DELETE
   ON S
   TO Jim,Fred,Mary;
```

1) A Name(SA3-"supplier authority three"- in this example)
2) One or more privileges (RETRIEVE – on certain attributes only – and DELETE, in the example)
3) The relvar to which the authority apply (relvar SA3 in the example) specified by means of the ON clause.
4) One or more "users" who are to be granted the specified privileges over the specified relvar, specified by means of the TO clause.

- The general Systax.

```
AUTHORITY  <AUTHORITY NAME>
   GRANT <PRIVILEGE COMMALIST>
      ON <relvar name>
      TO<user ID commalist>
```

- RETRIVE , INSERT , UPDATE & DELETE are self-explanatory. INSERT & UPDATE with a comma list of attributes names are define analogously. The specification ALL is shorthand for all privileges : RETRIEVE (all attribute) , INSERT (all attribute), UPDATE (all attribute) and DELETE.

Mandatory Access Control:-

- Mandatory control are applicable to database in which the data has a rather static and rigid classification structure. The basic idea is that each data object has a classification level and each user has clearance level.
  - User A can retrieve object B only if the clearance level of A is greater than or equal to the classification level of B (simple security property)
  - User A can update object B only if the clearance level of A is equal to the classification level of B (the "Star property")

- The first rule here is obvious enough , but the security requires a word of explanation
  - **Discretionary protection** : class c is divide into two subclass C1 & C2 (where C1 is less secure than C2). Each support discretionary controls, meaning that access is subject to the discretion of data owner.
  - **Mandatory protection**: class B is the class that deals with mandatory controls. Its is further divide into subclass B1,B2 & B3.
    - Class B1 requires "labelled security protection"
    - Class B2 additional requires a formal statement of the same thing. It also requires that convert channel be identified and eliminated.
    - Class B3 specifically requires audit and recovery support as well as a designated security administrator.
  - **Verified protection** : class A ,the most secure ,require a mathematical proof that the security mechanism is consist and that it is adequate to support the specified security policy.

[Que :- Short note on Statistical Database]

Statistical Database:-

- A statistical database is a database that permits queries that derived aggregate information (e.g sum, average) but not that derived individual information.
- The problem with such database is that sometimes it is possible to make inferences from legal queries to deduce the answer to illegal ones.
- The security of the database has clearly been compromised even though user U has issued only legitimate statistical queries.
- If the user can find a Boolean expression that identifier some individual , then information regarding that individual is no longer secure.
- This fact suggest that the system should refuse to respond to a query for which the cardinality of the set to be summarized is less then some one.
- Example:- query which contains SUM,MAX,MIN,AVG,COUNT function
  - **BANK TABLE**

Accunt No	Name	Total_amount
A001	Rakesh	30000
A002	Ashvin	22000
A003	Aryan	35000

- Retrieve AVG of Amount of Account Holder from BANK Table

  Select AVG (Total_amount) from BANK table;

[Que:- what is Data Encryption? Explain with example.]

Data Encryption:-

- The most effective countermeasure against such threats is – data Encryption – That is storing and transmitting sensitive data in encryption form .
- The original (unencrypted) data is called the plaintext.
- The plaintext is encrypted by subjecting it to an encryption algorithm. Whose input are the plaintext and an encryption Key , the output form this algorithm – the encryption form of the plaintext – is called the **Cipher text.**
- The Data Encryption Standard:-
  - The example above made use of a substitution procedure.
  - Substitution is one of the two basic approach to encryption as traditional practice.
- One such algorithm is the **Database Encryption Standard (DES) .**
  - To use the DES , plaintext is divided into 64 – bit blocks and each block is encrypted using a 64 – key.
- **Public Key Encryption :-**
  - In a public key scheme , both the encryption algorithm and the encryption key are made freely available , thus anyone can convert plaintext into cipher text .
  - But the corresponding decryption key is kept secret.

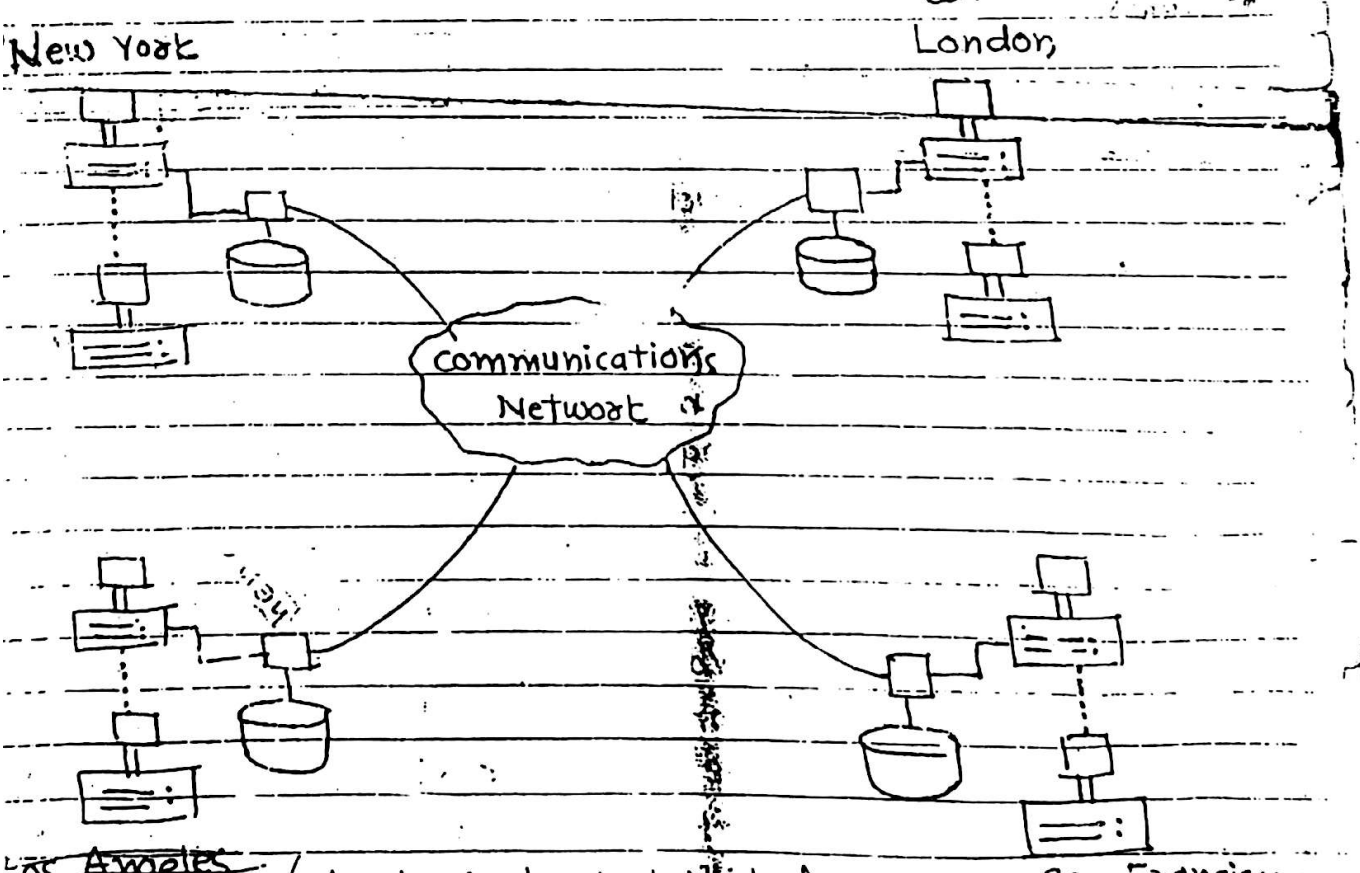
T.Y. B.C.A

KBHI

Distributed Databases

distributed database system consists of a collection of sites, connected together via some kind of communications networks, in which Each site is a full database system site in its own right. The sites have agreed to work together so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site.

Distributed database is really a kind of virtual database whose component parts are physically stored in a number of distinct real databases.



(A typical distributed Database System)

Each site is a database system site in its own right. other words, each site has its own local "real" database its own local users, its own local DBMS and transaction management software, and its own local data communications manager.
the distributed Database system can thus be regarded as a kind of partnership among the individual local DBMSs at the individual local sites.
Logically an extension of the local DBMS - provide the necessary partnership functionality, and it is the combination of this new component together with the existing DBMS that constitutes what is usually called the distributed database management system.

~~Advantages~~:
Distributed database is desirable because that enterprises are usually distributed already, at least logically and very likely physically too. Each organization unit within the enterprise will naturally maintain data that is relevent to its own operation.
It enables the structure of the database to mirror the structure of the enterprise - local data can be kept locally, where it most logically belongs - while at the same time remote data can be accessed when necessary.
In figure suppose there are only two sites, New York and London, and suppose system is a baking system, with account data for New York accounts kept in New York and account data for London accounts kept in London. Then the advantages are surely obvious. The distributed management combines efficiency of processing sibility and accessibility

Reliability :-
The probability that the system is up and running at any given moment.

Availability :-
The system is up and running continuously throughout a specified period.

(4) **Location Independence** :-
Users should not have to know where data is physically stored.
Location independence is desirable because it simplifies user programs and terminal activities.
It allows data to migrate from site to site without invalidating any of those programs or activities.
Such migratability is desirable because it allows data to be moved around the network in response to changing performance requirement .

(5) **Fragmentation Independence :-**
A system supports data fragmentation if a given relvar can be divided up into pieces or fragments for physical storage purposes.

Fragmentation is desirable for performance reasons :
Data can be stored at the location where it is most frequently used. So that most operations are local and network traffic is reduced.

For Example :-
Employee relvar EMP. In system that supports fragmentations, we might define two fragments as follows

FRAGMENT EMP AS
N_emp at sit 'New York' where dept# = 'D1' or dept # = 'D3'
L_emp at site 'LONDON' where dept# = 'D'

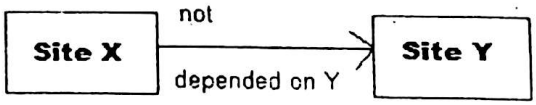
Que Which are the basic principle (OR Twelve Objects) of Distributed database system.

→ There are 12 basic principle. That are as under.....

1. Local Autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence

(1) Local Autonomy :-

The sites in a distributed system should be autonomous .
Local Autonomy means that all operations at a given site are controlled by that site.
No site X should depend on some other site Y for its successful operation.
Local Autonomy also implies that local data is locally owned and managed with local accountability.



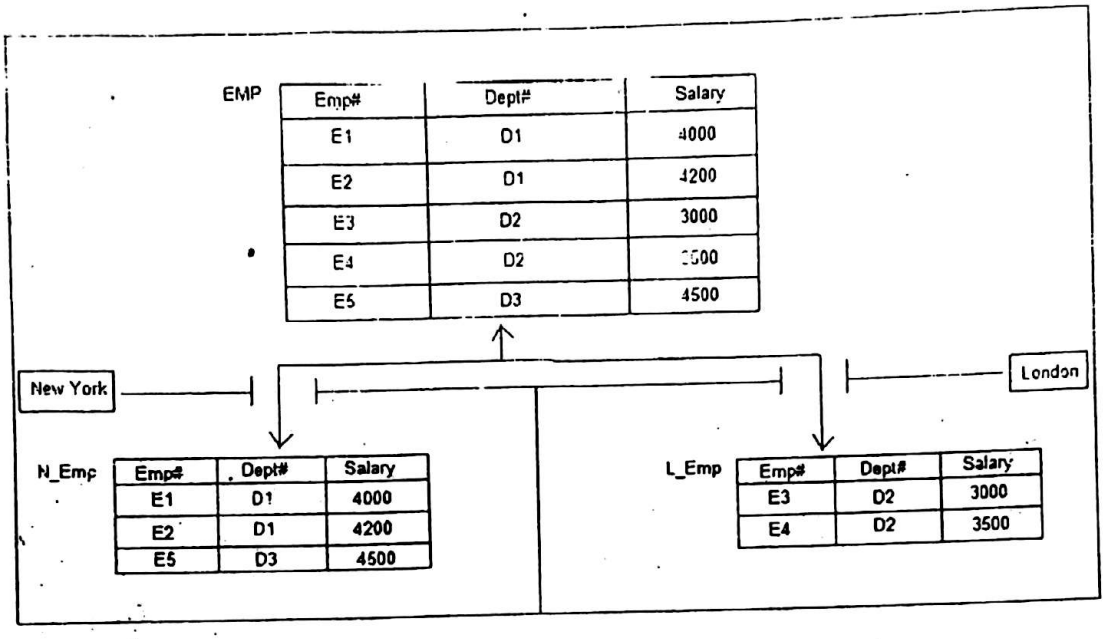
(2) No reliance on a central site :-

Local Autonomy implies that all sites must be treated as equals.
Therefore there must not be any reliance on a central 'master' site for some central service:
For example :-
Centralized query processing centralized transaction management or centralized naming services.
Such that the entire system is dependent on that central site.

Disadvantage :-
(1)Central site might be a bottleneck.
(2)If the central site went down, the whole system would be down.

(3) Continuous operation :-

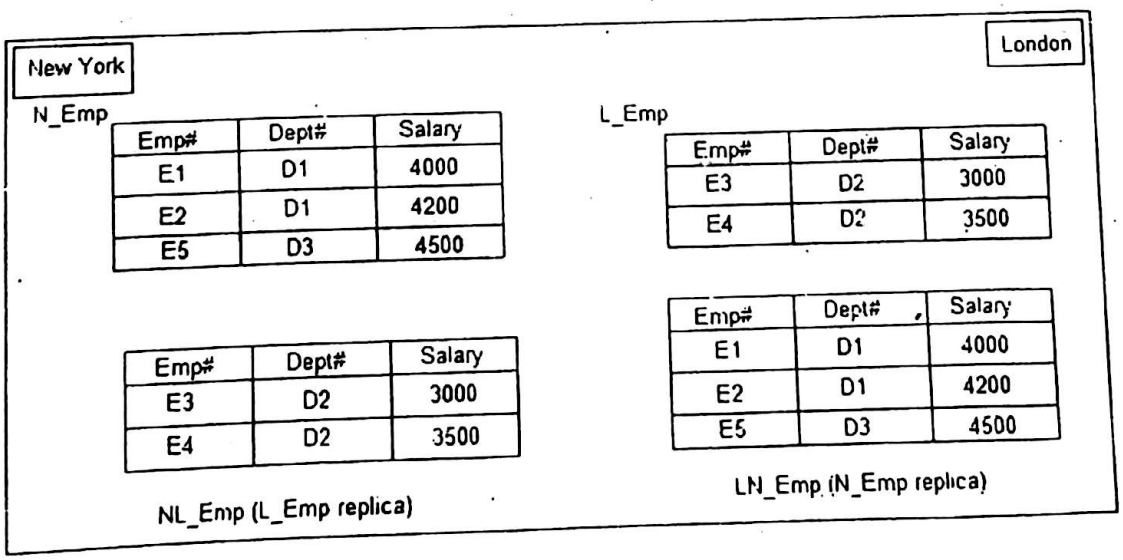
An advantage of distributed systems in general is that they should provide greater reliability and greater availability.



(6) Replication Independence :-

A system supports data replication if a given fragment of a given stored relvar can be represented by many distinct copies or replicas stored at many distinct sites.

Replicate N_emp as LN_emp at site 'LONDON';
Replicate L_emp as NL_emp at site 'NEW YORK';



Replication is desirable for at least two reasons :

(1) It can mean better performance.

(2) It can also mean better availability. Because at least one copy remains available.

Disadvantages :-
When a given replicated object is updated, all copies of that object must be updated
The update propagation problem.

(7) **Distributed query processing :-**
Consider the query "Get London supplies at red parts".
Suppose the user is at the New York site and the data is at the London site.
Suppose too that there are N suppliers that satisfy the request.
The query will basically involve two messages one to send the request from New York to London, and one to return the result set of N tuples from London to New York.

(8) **Distributed transaction management :-**
There are two major aspects to transaction management recovery control & concurrency control.
In distributed system a single transaction can involve the execution of code at many sites.
It can involve updates at many sites each transaction is therefore said to consist of several agents.
Where an agent is the process performed in behalf of a given transaction at a given site.
Two agents that are part of the same transaction must obviously not be allowed to deadlock with each other.

(9) **Hardware independence :-**
It is desirable to be able to run the same DBMS on different hardware platforms And therefore different machines all participate as equal partner in a distributed system.

(10) **Operating system independence :-**
It is able to run the same DBMS on different operating system platforms.

(11) **Network independence :-**
It is able to support a variety of disparate communication networks.

(12) **DBMS independence :-**
They often run different DBMS and it would be nice if those different DBMS could all participate in a distributed system.
In other word, the ideal distributed system should provide DBMS independence.