14CS IT 504

Hall Ticket Number:

III/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

November, 2017 **Common for CSE & IT**

Fifth Semester **DATABASE MANAGEMENT SYSTEMS**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Answer all questions (1X12=12 Marks)

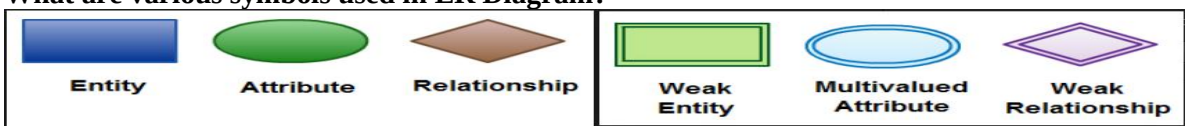
a) **What is Data Integrity and Security?**
Data integrity and data security are two different aspects that make sure the usability of data is preserved all the time. Main difference between integrity and security is that integrity deals with the validity of data, while security deals with protection of data.

b) **What is the role of Database application programmers ?**
Perform database programming for new and existing systems. Write scripts, stored procedures and functions for database system. Perform quality assurance tests for ensuring data integrity and quality.

c) **Define degree of a relation**.
The degree of a relation is the number of attributes n of its relation schema.

d) **Define attribute and entity set.**
Attribute represents some property of interest that further describes an entity, such as the employee's name or salary.
Entity set: The set of all entities of the same type is termed as an entity set

e) **What are various symbols used in ER Diagram?**



f) **What is file organization?**
A file organization refers to the organization of the data of a file into records, blocks, and access structures.

g) **Differentiate between B-tree and B+ tree.**
1.In a B tree search keys and data stored in internal or leaf nodes. But in B+-tree data store only leaf nodes.
2.Searching any data in a B+ tree is very easy because all data are found in leaf nodes. In a B tree, data cannot be found in leaf nodes.
3.In a B tree, data may be found in leaf nodes or internal nodes. Deletion of internal nodes is very complicated. In a B+ tree, data is only found in leaf nodes. Deletion of leaf nodes is easy.
4. Insertion in B tree is more complicated than B+ tree.
5. B+ trees store redundant search key but B tree has no redundant value

h) **Define Durability.**
Durability or permanency. The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

i) **What is functional dependency?**
A functional dependency, denoted by X → Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y] = t2[Y].

j) **Define Static Hashing.**
Hashing is used as an internal search structure within a program whenever a group of records is accessed exclusively by using the value of one field.

k) **What is binary lock?**
A binary lock can have two states or values: locked and unlocked (or 1 and 0, for simplicity). A distinct lock is associated with each database item X.

l) **What is Concurrency?** .
Concurrency is the ability of a database to allow multiple users to affect multiple transactions.

2. a) **Describe different types of database models. Compare and contrast their features.** 6M

Types of Data Models-----4M

A data model instance may be one of three kinds.
Conceptual data model : Describes the semantics of a domain, being the scope of the model. For example, it may be a model of the interest area of an organization or industry. This consists of entity classes, representing kinds of things of significance in the domain, and relationship assertions about associations between pairs of entity classes. A conceptual schema specifies the kinds of facts or propositions that can be expressed using the model. In that sense, it defines the allowed expressions in an artificial 'language' with a scope that is limited by the scope of the model.
Logical data model : Describes the semantics, as represented by a particular data manipulation technology. This consists of descriptions of tables and columns, object oriented classes, and XML tags, among other things.
Physical data model : Describes the physical means by which data are stored. This is concerned with partitions, CPUs, tablespaces, and the like.

Compare and contrast their features.---2M

Here we compare these three types of data models. The table below compares the different features:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	✓
Foreign Keys		✓	✓
Table Names			✓
Column Names			✓
Column Data Types			✓

b) **Discuss the roles of various actors on the scene and workers behind the screen.** 6M

Actors on the scene--3M

Database Administrator (DBA)
In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the

database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

Database designers

They are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

End users

They are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

■ Casual end users occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

■ Naive or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions.

Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

■ Standalone users maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.

Workers behind the scene--3M

DBMS system designers and implementers design and implement the DBMS modules and interfaces as a software package. A DBMS is a very complex software system that consists of many components, or modules, including modules for implementing the catalog, query language processing, interface processing, accessing and buffering data, controlling concurrency, and handling data recovery and security.

Tool developers design and implement tools—the software packages that facilitate database modeling and design, database system design, and improved performance. Tools are optional packages that are often purchased separately.

Operators and maintenance personnel (system administration personnel) are responsible for the actual running and maintenance of the hardware and software environment for the database system.

OR

3. a) Explain Centralized and Client/Server Architectures for DBMSs with neat diagrams. 8M

Centralized DBMSs Architecture ---3M

Earlier architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality. The reason was that most users accessed such systems via computer terminals that did not have processing power and only provided display capabilities. Therefore, all processing was performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.

At first, database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a centralized DBMS in which all the DBMS functionality, application program execution, and user inter-face processing were carried out on one machine. Figure illustrates the physical components in a centralized architecture. Gradually, DBMS systems

started to exploit the available processing power at the user side, which led to client/server DBMS architectures.

Diagram--1M

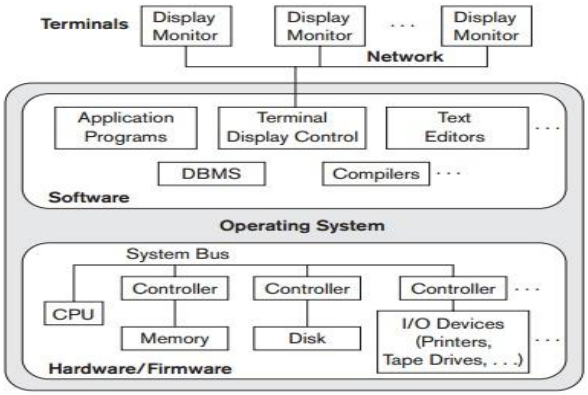


Figure 2.4
A physical centralized architecture.

2. Basic Client/Server Architectures-----3M

The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network. The idea is to define specialized servers with specific functionalities. For example, it is possible to connect a number of PCs or small workstations as clients to a file server that maintains the files of the client machines. Another machine can be designated as a printer server by being connected to various printers; all print requests by the clients are forwarded to this machine. Web servers or e-mail servers also fall into the specialized server category. The resources provided by specialized servers can be accessed by many client machines. Theclient machines provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

Diagram--1M

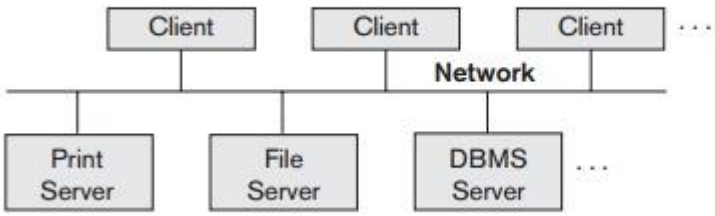


Figure 2.5
Logical two-tier client/server architecture.

b) Write a short notes on weak entity? 4M

Any relevant explanation----4M

A weak entity type normally has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity. If we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the partial key. In the worst case, a composite attribute of all the weak entity's attributes will be the partial key. Weak entity types can sometimes be represented as complex (composite, multivalued) attributes. In the preceding example, we could specify a multivalued attribute. Dependents for EMPLOYEE, which is a composite attribute with component attributes Name, Birth_date, Sex, and Relationship. The choice of which representation to use is made by the database designer. If the weak entity participates independently in relationship types other than its identifying relationship type, then it should not be modeled as a complex attribute.

UNIT-II

4. a) Write the queries in relational algebra

Employee(Fname, Minit, Lname, Ssn, Address, Bdate,Sex, Salary, Super_ssn, Dno)
Department(Dname, Dno, Mgr_ssn, Mgr_start_date)
Dept_location(Dnumber, Dlocation)
Project(Pname, Pnumber, Plocation, Dnum) **6M**
Works_on(Essn, Pno, Hours)
Dependent(Essn, Dependent_name,Sex, Bdate, Relationship)

- **Find the names of employees who work on all the projects controlled by department number 5.**
  - **Retrieve the name and address of all employees who work for the Research department.**
  - **List the names of all employees with two or more dependents.**

Each Query carries 2M

select fname from employees, projects,works_on where project.pnumber=works_on.pno and works_on.essn=employee.ssn and project.dnum=5

select fname, address from employee, department where employee.dno=department.dno and department.dname= 'RESEARCH'.

select fname from dependent_name,employee where employee.ssn=dependent.essn and count(dependent_name)>2.

b) Discuss various Schema Change Statements in SQL along with examples. 6M

Any relevant explanation---3M

The DROP command can be used to drop named schema elements, such as tables, domains, or constraints. One can also drop a schema. For example, if a whole schema is no longer needed, the DROP SCHEMA command can be used. There are two drop behavior options: CASCADE and RESTRICT. For example, to remove the COMPANY database schema and all its tables, domains, and other elements, the CASCADE option is used as follows:
DROP SCHEMA COMPANY CASCADE;
If the RESTRICT option is chosen in place of CASCADE, the schema is dropped only if it has no elements in it; otherwise, the DROP command will not be executed. To use the RESTRICT option, the user must first individually drop each element in the schema, then drop the schema itself. If a base relation within a schema is no longer needed, the relation and its definition can be deleted by using the DROP TABLE command. For example, if we no longer wish to keep track of dependents of employees in the COMPANY database we can get rid of the DEPENDENT relation by issuing the following command:
DROP TABLE DEPENDENT CASCADE;
If the RESTRICT option is chosen instead of CASCADE, a table is dropped only if it is not referenced in any constraints.

The ALTER Command **Any relevant explanation---3M**
The definition of a base table or of other named schema elements can be changed byusing the ALTER command. For base tables, the possible alter table actions include adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints. For example, to add an attribute for keeping track of jobs of employees to the EMPLOYEE base relation in the COMPANY schema we can use the command
ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job VARCHAR(12);

We must still enter a value for the new attribute Job for each individual EMPLOYEE tuple. This can be done either by specifying a default clause or by using the UPDATE. To drop a column, we must choose either CASCADE or RESTRICT for drop behavior. If CASCADE is chosen, all constraints and views that reference the column are dropped automatically from the schema, along with the column. If RESTRICT is chosen, the command is successful only if no views or

constraints reference the column. For example, the following command removes the attribute Address from the EMPLOYEE base table:

ALTER TABLE COMPANY.EMPLOYEE DROP COLUMN Address CASCADE;

It is also possible to alter a column definition by dropping an existing default clause or by defining a new default clause. The following examples illustrate this clause:

ALTER TABLE COMPANY.DEPARTMENT ALTER COLUMN Mgr_ssn
DROP DEFAULT;
ALTER TABLE COMPANY.DEPARTMENT ALTER COLUMN Mgr_ssn
SET DEFAULT '333445555';

OR

5 a) Consider the following database relations. Write SQL statements given below:

S (S#, SNAME, SCITY) P (P#, PNAME, PCITY)
J (J#, JNAME, JCITY) SPJ (S#, P#, J#, QTY)

i)Get J# values for projects using one part available for supplier?
ii)Get P# values for part supplied to any project in London by a London supplier.
iii)Get JNAME for projects supplied by at least one supplier not in the same city.

Each Query carries 2M

```
i) SELECT J.JNO FROM J,S,SPJ WHERE J.JNO=SPJ.JNO AND S.SNO=SPJ.SNO AND
   COUNT(S.SNO)=1

ii) SELECT DISTINCT P.PNO  FROM  P,S,J,SPJ WHERE P.PNO = SPJ.PNO AND
 S.SNO = SPJ.SNO AND  J.JNO = SPJ.JNO AND  S.CITY = 'LONDON' AND  J.CITY = 'LONDON';

iii) SELECT J.JNAME  FROM S,J,SPJ  WHERE S.SNO = SPJ.SNO AND J.JNO = SPJ.JNO AND
 S.CITY <> J.CITY;
```

b) Write short notes on Correlated Nested Queries and Aggregate Functions in SQL. 6M

Writing any relevant points can be considered --3M

Correlated Nested Queries

Whenever a condition in the WHERE clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated. We can understand a correlated query better by considering that the nested query is evaluated once for each tuple (or combination of tuples) in the outer query. For example, For each EMPLOYEE tuple, evaluate the nested query, which retrieves the Essn values for all DEPENDENT tuples with the same sex and name as that EMPLOYEE tuple; if the Ssn value of the EMPLOYEE tuple is in the result of the nested query, then select that EMPLOYEE tuple. In general, a query written with nested select-from-where blocks and using the = or IN comparison operators can always be expressed as a single block query.

Aggregate Functions

Writing any relevant points can be considered --3M

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary. Grouping is used to create subgroups of tuples before summarization. Grouping and aggregation are required in many database applications, and we will introduce their use in SQL through examples.

A number of built-in aggregate functions exist: COUNT, SUM, MAX, MIN, and AVG. The COUNT function returns the number of tuples or values as specified in a The functions SUM, MAX, MIN, and AVG can be applied to a set or multiset of numeric values and return, respectively, the sum, maximum value, minimum value, and average (mean) of those values. These functions can be used in the SELECT clause or in a HAVING clause (which we introduce later). The functions MAX and MIN can also be used with attributes that have nonnumeric domains if the

domain values have a total ordering among one another. We illustrate the use of these functions with sample queries.

Query 1. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.

Q1: Select SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary) from Employee;

UNIT III

6 a) What are the differences between Files of Unordered records and ordered records? 4M

Writing any relevant points can be considered

In an ordered file, the records are sequenced on some field in the record (like StudentId or StudentName etc). In an unordered file, the records are not in any particular order.

Ordered File	Unordered File
RecordA	RecordM
RecordB	RecordH
RecordG	RecordB
RecordH	RecordN
RecordK	RecordA
RecordM	RecordK
RecordN	RecordG

Records are placed in the file in the order in which they are inserted, so new records are inserted at the end of the Such an organization is called a heap or pile file. This organization is often used with additional access paths, such as the secondary indexes. Inserting a new record is very efficient. The last disk block of the file is copied into a buffer, the new record is added, and the block is then rewritten back to disk. The address of the last file block is kept in the file header. However, searching for a record using any search condition involves a linear search through the file block by block an expensive procedure.

To delete a record, a program must first find its block, copy the block into a buffer, delete the record from the buffer, and finally rewrite the block back to the disk. This leaves unused space in the disk block. Deleting a large number of records in this way results in wasted storage space.

An ordered Sequential file, is a file ordered upon some key field. The ordering of the records in the file makes it possible to process an ordered file in ways that are not available to us with unordered files. While it is not really possible to apply batch updates or deletes to an unordered file these are possible with ordered files. Ordered records have some advantages over unordered files. First, reading the records in order of the ordering key values becomes extremely efficient because no sorting is required. Second, finding the next record from the current one in order of the ordering key usually requires no additional block accesses

b) Discuss various informal design guidelines for Relation Schemas. 8M

Each Guideline explanation--2M

Informal guidelines that may be used as measures to determine the quality of relation schema design:

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Making sure that the semantics of the attributes is clear in the schema

Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation. Intuitively, if a relation schema corresponds to one entity type or one relationship type, it is straightforward to interpret and to explain its meaning. Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

Reducing the redundant information in tuples

Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present,4 note them clearly and make sure that the programs that update the database will operate correctly.

Reducing the NULL values in tuples

As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation. Using space efficiently and avoiding joins with NULL values are the two overriding criteria that determine whether to include the columns that may have NULLs in a relation or to have a separate relation for those columns.

Disallowing the possibility of generating spurious tuples

Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

OR

7 a) Explain briefly about buffering of blocks 6M

When several blocks need to be transferred from disk to main memory and all the block addresses are known, several buffers can be reserved in main memory to speed up the transfer. While one buffer is being read or written, the CPU can process data in the other buffer because an independent disk I/O processor (controller) exists that, once started, can proceed to transfer a data block between memory and disk independent of and in parallel to CPU processing.

Figure 1 illustrates how two processes can proceed in parallel. Processes A and B are running concurrently in an interleaved fashion, whereas processes C and D are running concurrently in a parallel fashion. When a single CPU controls multiple processes, parallel execution is not possible. However, the processes can still run concurrently in an interleaved way. Buffering is most useful when processes can run concurrently in a parallel fashion, either because a separate disk I/O processor is available or because multiple CPU processors exist.

Figure 2 illustrates how reading and processing can proceed in parallel when the time required to process a disk block in memory is less than the time required to read the next block and fill a buffer. The CPU can start processing a block once its transfer to main memory is completed; at the same time, the disk I/O processor can be reading and transferring the next block into a different buffer. This technique is called double buffering and can also be used to read a continuous stream of blocks from disk to memory. Double buffering permits continuous reading or writing of data on consecutive disk blocks, which eliminates the seek time and rotational delay for all but the first block transfer. Moreover, data is kept ready for processing, thus reducing the waiting time in the programs.

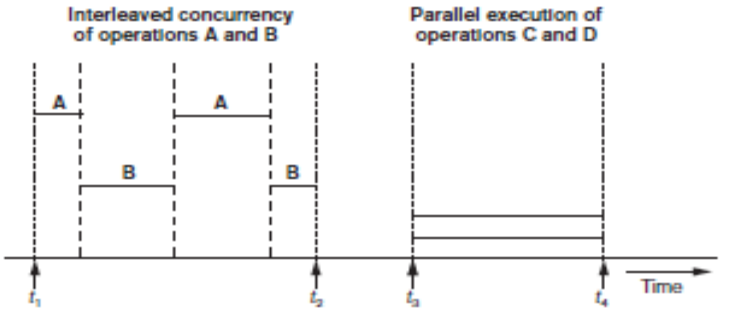


Figure 1
Interleaved concurrency versus parallel execution.

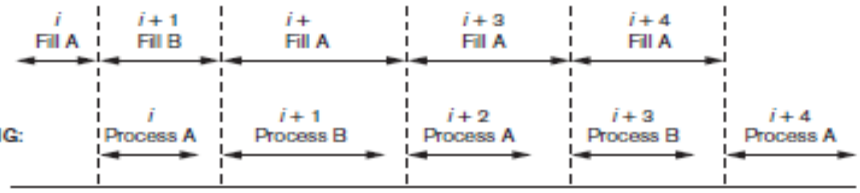


Figure 2
Use of two buffers, A and B, for reading from disk.

b) Define Boyce-Codd Normal Form. Explain it by demonstrating with an example. **6M**

Definition--3M
Explanation with any example---3M

Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF. That is, every relation in BCNF is also in 3NF; however, a relation in 3NF is *not necessarily* in BCNF.

Definition. A relation schema *R* is in **BCNF** if whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in *R*, then *X* is a superkey of *R*.

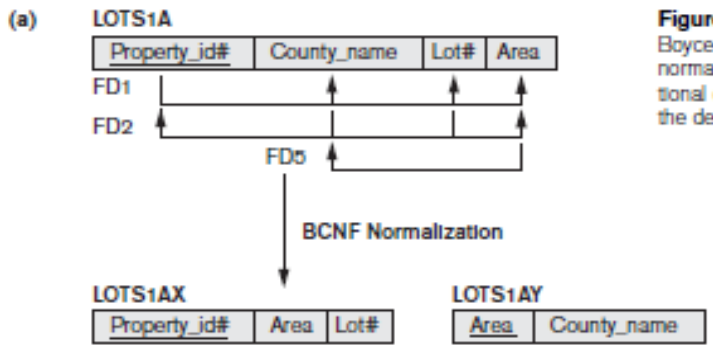


Figure 1
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition.

UNIT IV

8 a) Explain Two-Phase Locking Techniques for Concurrency Control. **8M**

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it.

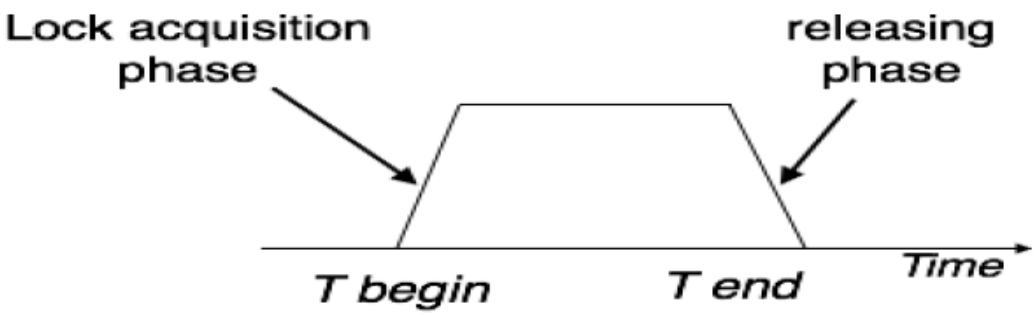
Locks are of two kinds .

 Binary Locks − A lock on a data item can be in two states; it is either locked or unlocked.

 Shared/exclusive − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

Two-Phase Locking 2PL

 This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.

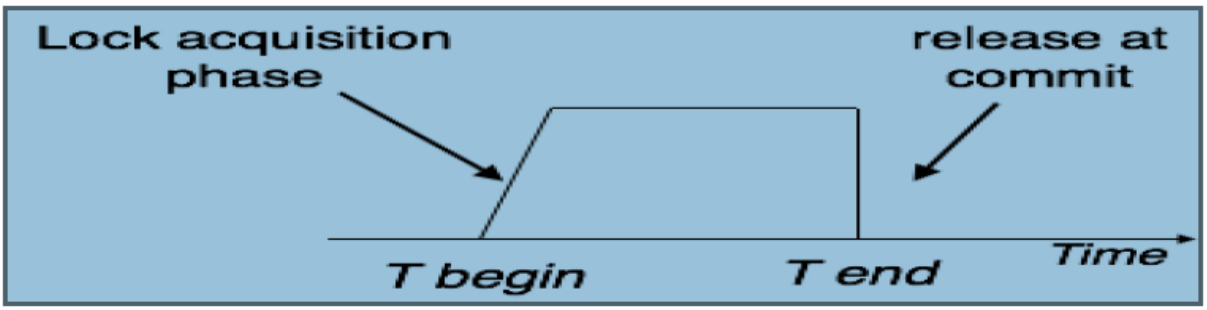


 Two-phase locking has two phases, one is **growing**, where all the locks are being acquired by the transaction; and the second phase is **shrinking,** where the locks held by the transaction are being released.

 To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking

 The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



b) Discuss different types of failures. 4M

Types of Failures. Failures are generally classified as transaction, system, and media failures. There are several possible reasons for a transaction to fail in the middle of execution:

1. A computer failure (system crash). A hardware, software, or network error occurs in the computer system during transaction execution. Hardware crashes are usually media failures—for example, main memory failure.

2. A transaction or system error. Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous

parameter values or because of a logical programming error.3 Additionally, the user may interrupt the transaction during its execution.

3. Local errors or exception conditions detected by the transaction. During transaction execution, certain conditions may occur that necessitate cancellation of the transaction. For example, data for the transaction may not be found. An exception condition,4 such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be canceled. This exception could be programmed in the transaction. itself, and in such a case would not be considered as a transaction failure.

4. Concurrency control enforcement. The concurrency control method may decide to abort a transaction because it violates serializability or it may abort one or more transactions to resolve a state of deadlock among several transactions

Transactions aborted because of serializability violations or deadlocks are typically restarted automatically at a later time.

5. Disk failure. Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.

6. Physical problems and catastrophes. This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

OR

9. a) What is a schedule? Explain about characterizing schedules based on recoverability. 6M

Schedule Definition---1M
Recoverability Explanation--5M

Schedule Definition

When transactions are executing concurrently in an interleaved fashion, then the order of execution of operations from all the various transactions is known as a
schedule (or history).

Characterizing schedules based on recoverability

For some schedules it is easy to recover from transaction and system failures, whereas for other schedules the recovery process can be quite involved. In some cases, it is even not possible to recover correctly after a failure. Hence, it is important to characterize the types of schedules for which recovery is possible, as well as those for which recovery is relatively simple.

First, we would like to ensure that, once a transaction T is committed, it should never be necessary to roll back T. This ensures that the durability property of transactions is not violated. The schedules that theoretically meet this criterion are called recoverable schedules; those that do not are called non recoverable and hence should not be permitted by the DBMS. The definition of **recoverable schedule** is as follows: A schedule S is recoverable if no transaction T in S commits until all transactions T_ that have written some item X that T reads have committed. A transaction T reads from transaction T_ in a schedule S if some item X is first written by T_ and later read by T. In addition, T_ should not have been aborted before T reads item X, and there should be no transactions that write X after T_ writes it and before T reads it (unless those transactions, if any, have aborted\ before T reads X).

In a recoverable schedule, no committed transaction ever needs to be rolled back, and so the definition of committed transaction as durable is not violated. However, it is possible for a phenomenon known as **cascading rollback** (or cascading abort) to occur in some recoverable schedules, where an uncommitted transaction has to be rolled back because it read an item from a transaction that failed.

A schedule is said to be **cascadeless, or to avoid cascading rollback**, if every transaction in the schedule reads only items that were written by committed transactions. In this case, all items read will not be discarded, so no cascading rollback will occur.

Finally, there is a third, more restrictive type of schedule, called a **strict schedule**, in which transactions can neither read nor write an item X until the last transaction that wrote X has committed (or aborted).

b) Discuss about Recovery techniques based on Immediate Update **6M**

Recovery Techniques Based on Immediate Update **[Explanation--6M]**

In the immediate update techniques, the database may be updated by the operations of a transaction immediately, before the transaction reaches its commit point. However, these operations are typically recorded in the log on disk by force writing before they are applied to the database so that recovery is possible.

When immediate update is allowed, provisions must be made for undoing the effect of update operations on the database, because a transaction can fail after it has applied some updates to the database itself. Hence recovery schemes based on immediate update must include the capability to roll back a transaction by undoing the effect of its write operations.

1. When a transaction starts, write an entry start_transaction(T) to the log;

2. When any operation is performed that will change values in the database, write a log entry write_item(T, x, old_value, new_value);

3. Write the log to disk;

4. Once the log record is written, write the update to the database buffers;

5. When convenient write the database buffers to the disk;

6. When a transaction is about to commit, write a log record of the form commit(T);

7. Write the log to disk.

The protocol and how different entries are affected can be best summarised in Figure below.

.

Log entry	Log written to disk	Changes written to database buffer	Changes written on disk
start_transaction(T)	No	N/A	N/A
read_item(T, x)	No	N/A	N/A
write_item(T, x)	Yes	Yes	*Yes
commit(T)	Yes	Undefined	Undefined
Checkpoint	Yes	Undefined	Yes(of committed Ts)

*Yes: writing back to disk may not occur immediately

Scheme prepared by **Signature of HOD, IT Dept**

Paper Evaluators:

Sno	Name of the college	Name of the examiner	Signature