14CS IT 504

Hall Ticket Number:

III/IV B.Tech (Supplementary) DEGREE EXAMINATION

April, 2018 **Common for CSE & IT**

Fifth Semester **DATABASE MANAGEMENT SYSTEMS**

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

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

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

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

a) Define Database and DBMS.
A database is a collection of related data.By data,we mean known facts that can be recorded and that have implicit meaning.
It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

b) What is Database schema?
Database Schema is the overall Design of the Database. It is the skeleton structure that represents the logical view of the entire database. It tells how the data is organized and how the relations among them are associated.

c) Define primary key and foreign key.
A primary key, also called a primary keyword, is akey in a relational database that is unique for each record.
A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table or the same table.[1][2][3] In simpler words, the foreign key is defined in a second table, but it refers to the primary key or a unique key in the first table.

d) What is recursive relationship?
If the same entity type participate more than once in a relationship type in different roles then such relationship types are called recursive relationship.
OR
A recursive relationship is a relationship between an entity and itself.

e) What is database catalog?
The system catalog is a vital part of a database. Inside the database, there are objects, which include tables, views and indexes. Basically, the system catalog is a set of objects, which includes information that defines the database structure itself.

f) Write the basic form of Select statement?
SELECT *column1*, *column2*, ... FROM *table_name*;

g) Define DDL.
It is a language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionary and tables within databases.

h) Define Second Normal Form.
A relation R is in second normal form (2NF) if and only if it is in 1NF and every nonkey attribute is fully functionally dependent on the primary key.

i) What is functioning of system log?

j) Define lock
A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database.

k) When a schedule is said to be cascadeless?
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.

l) Define timestamp.
Timestamp is a unique identifier created by the DBMS to identify a transaction. Typically, timestamp values are assigned in the order in which the transactions are submitted to the system, so a timestamp can be thought of as the transaction start time. We will refer to the timestamp of transaction T as TS(T).

UNIT I

2. a) Discuss the main characteristics of the database approach and how it differs from traditional file systems. 6M

The main characteristics of the database approach versus the file-processing approach are the followin

■ **Self-describing nature of a database system**
■ **Insulation between programs and data, and data abstraction**
■ **Support of multiple views of the data**
■ **Sharing of data and multiuser transaction processing**

Listing the characterist
Writing any relevant poi

Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only th itself but also a complete definition or description of the database structure and constraints.
In traditional file processing, data definition is typically part of the application programs themselv these programs are constrained to work with only one specific database, whose structure is decla application programs.

Insulation between Programs and Data and Data Abstraction

In traditional file processing, the structure of data files is embedded in the application programs, so a to the structure of a file may require changing all programs that access that file. By contrast, DB programs do not require such changes in most cases. The structure of data files is stored in the DB separately from the access programs. We call this property program-data independence. In som database systems, such as object-oriented and object-relational systems , users can define operations part of the database definitions. An operation is specified in two parts. The interface of an operation i operation name and the data types of its arguments. The implementation (or method) of the operation i separately and can be changed without affecting the interface. User application programs can operate by invoking these operations through their names and arguments, regardless of how the oper implemented. This may be termed program-operation independence.

Support of Multiple Views of the Data

A database typically has many users, each of whom may require a different perspective or view of th A view may be a subset of the database or it may contain virtual data that is derived from the database not explicitly stored. A multiuser DBMS whose users have a variety of distinct applications mu facilities for defining multiple views. For example, one user of the database may be interested only in and printing the transcript of each student; A second user, who is interested only in checking that stu taken all the prerequisites of each course for which they register.

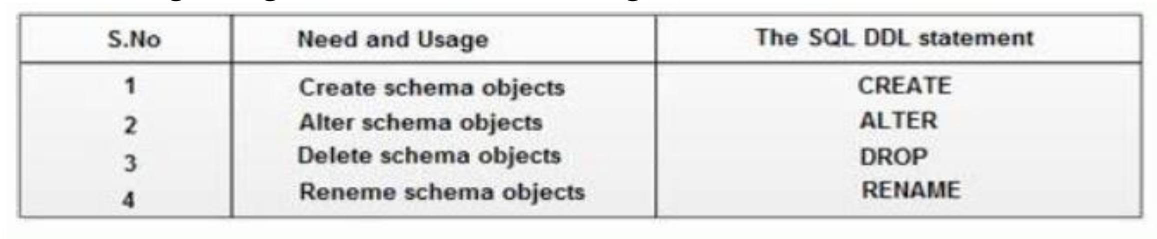
Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.

b) Write notes on database languages and Interfaces. 6M

Data Definition Language (DDL) **[DDL--2M]**

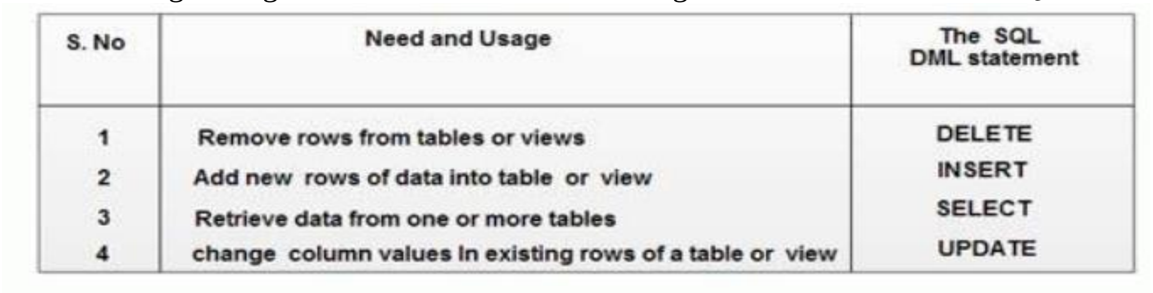
It is a language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionary and tables within databases.
It is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for each table and physical storage structure of each table on disk.
The following table gives an overview about usage of DDL statements in SQL



S.No	Need and Usage	The SQL DDL statement
1	Create schema objects	CREATE
2	Alter schema objects	ALTER
3	Delete schema objects	DROP
4	Reneme schema objects	RENAME

Data Manipulation Language (DML) **[DML--2M]**

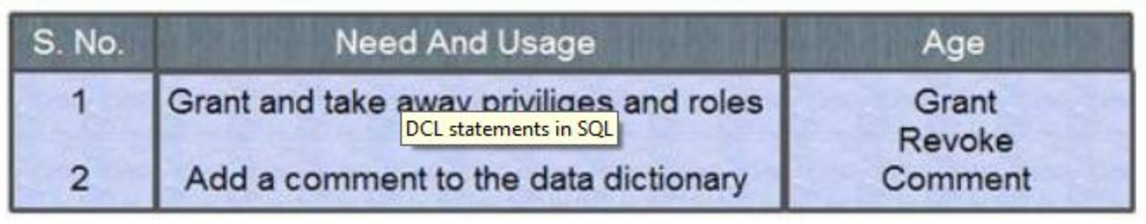
It is a language that provides a set of operations to support the basic data manipulation operations on the data held in the databases. It allows users to insert, update, delete and retrieve data from the database. The part of DML that involves data retrieval is called a query language.
The following table gives an overview about the usage of DML statements in SQL:



S. No	Need and Usage	The SQL DML statement
1	Remove rows from tables or views	DELETE
2	Add new rows of data into table or view	INSERT
3	Retrieve data from one or more tables	SELECT
4	change column values In existing rows of a table or view	UPDATE

Data Control Language (DCL) **[DCL--2M]**

DCL statements control access to data and the database using statements such as GRANT and REVOKE. A privilege can either be granted to a User with the help of GRANT statement. The privileges assigned can be SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX etc. In addition to granting of privileges, you can also revoke (taken back) it by using REVOKE command.
The following table gives an overview about the usage of DCL statements in SQL:



S. No.	Need And Usage	Age
1	Grant and take away priviliges and roles	Grant Revoke
2	Add a comment to the data dictionary	Comment

DCL statements in SQL

(OR)

3. a) Explain the main phases of database design with a neat diagram. 6M

[Explanation---4M]

The first step shown is **requirements collection and analysis**. During this step, the database interview prospective database users to understand and document their **data requirements**. ⁊ of this step is a concisely written set of users' requirements. These requirements should be sp as detailed and complete a form as possible. In parallel with specifying the data requirem useful to specify the known **functional requirements** of the application. These consi userdefined **operations** (or **transactions**) that will be applied to the database, including both and updates. In software design, it is common to use *data flow diagrams*, *sequence* *scenarios*, and other techniques to specify functional requirements. We will not discuss any techniques here; they are usually described in detail in software engineering texts. requirements have been collected and analyzed, the next step is to create a **conceptual schen** database, using a high-level conceptual data model. This step is called **conceptual des** conceptual schema is a concise description of the data requirements of the users and include descriptions of the entity types, relationships, and constraints; these are expressed using the provided by the high-level data model.

The next step in database design is the actual implementation of the database, using a co DBMS. Most current commercial DBMSs use an implementation data model—such as the rel the object-relational database model—so the conceptual schema is transformed from the high- model into the implementation data model. This step is called logical design or data model ma result is a database schema in the implementation data model of the DBMS.

The last step is the physical design phase, during which the internal storage structu organizations, indexes, access paths, and physical design parameters for the database files are In parallel with these activities, application programs are designed and implemented as transactions corresponding to the high level transaction specifications .

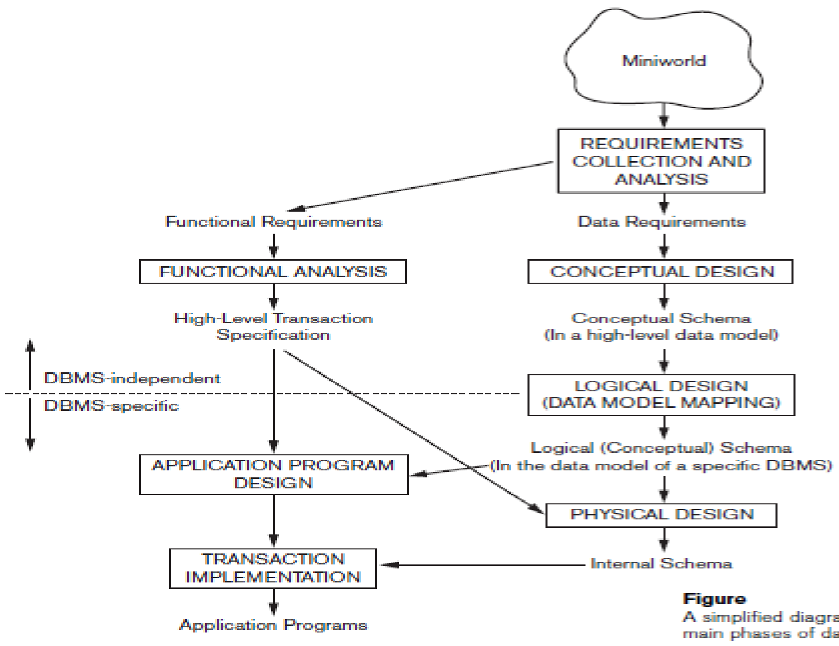

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure
A simplified diagram to illustrate the main phases of database design.

b) Discuss about various constraints on Relationship Types. 6M

[Any 4 Constraints explanation ---6M]

- **Domain Constraints**
- **Key Constraints**
- **Single Value Constraints**
- **Entity Integrity Constraint**
- **Referential Integrity Constraint**

Domain Constraints
Domain Constraints specifies that what set of values an attribute can take. Value of each attribute X must be an atomic value from the domain of X. The data type associated with domains include integer, character, string, date, time, currency etc. An attribute value must be available in the corresponding domain. Consider the example below .

Key Constraints
Keys are attributes or sets of attributes that uniquely identify an entity within its entity set. An Entity set E can have multiple keys out of which one key will be designated as the primary key. Primary Key must have unique and not null values in the relational table. In an subclass hierarchy, only the root entity set has a key or primary key and that primary key must serve as the key for all entities in the hierarchy.

Single Value Constraints
Single value constraints refers that each attribute of an entity set has a single value. If the value of an attribute is missing in a tuple, then we cal fill it with a "null" value. The null value for a attribute will specify that either the value is not known or the value is not applicable.

Entity Integrity Constraint
The Integrity Rule 1 is also called Entity Integrity Rule or Constraint. This rule states that no attribute of primary key will contain a null value. If a relation have a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained.

Referential Integrity Constraint
The integrity Rule 2 is also called the Referential Integrity Constraints. This rule states that if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

UNIT II

4. a) Explain the Unary and Set operations of relational algebra with examples 8M

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

Select
Project
Union
Set different
Cartesian product
Rename

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − σp(r)
Where σ stands for selection predicate and r stands for relation. p is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example **[Any relevant example]**
σsubject = "database"(Books)

σsubject = "database" and price = "450"(Books)

Project Operation (Π)
It projects column(s) that satisfy a given predicate.

Notation − ΠA1, A2, An (r)

Where A1, A2 , An are attribute names of relation r. Duplicate rows are automatically eliminated, as relation is a set.

For example – **[Any relevant example]**

Πsubject, author (Books)
Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪) **[Any relevant example]**
It performs binary union between two given relations and is defined as −

r ∪ s = { t | t ∈ r or t ∈ s}

Notation − r U s
Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

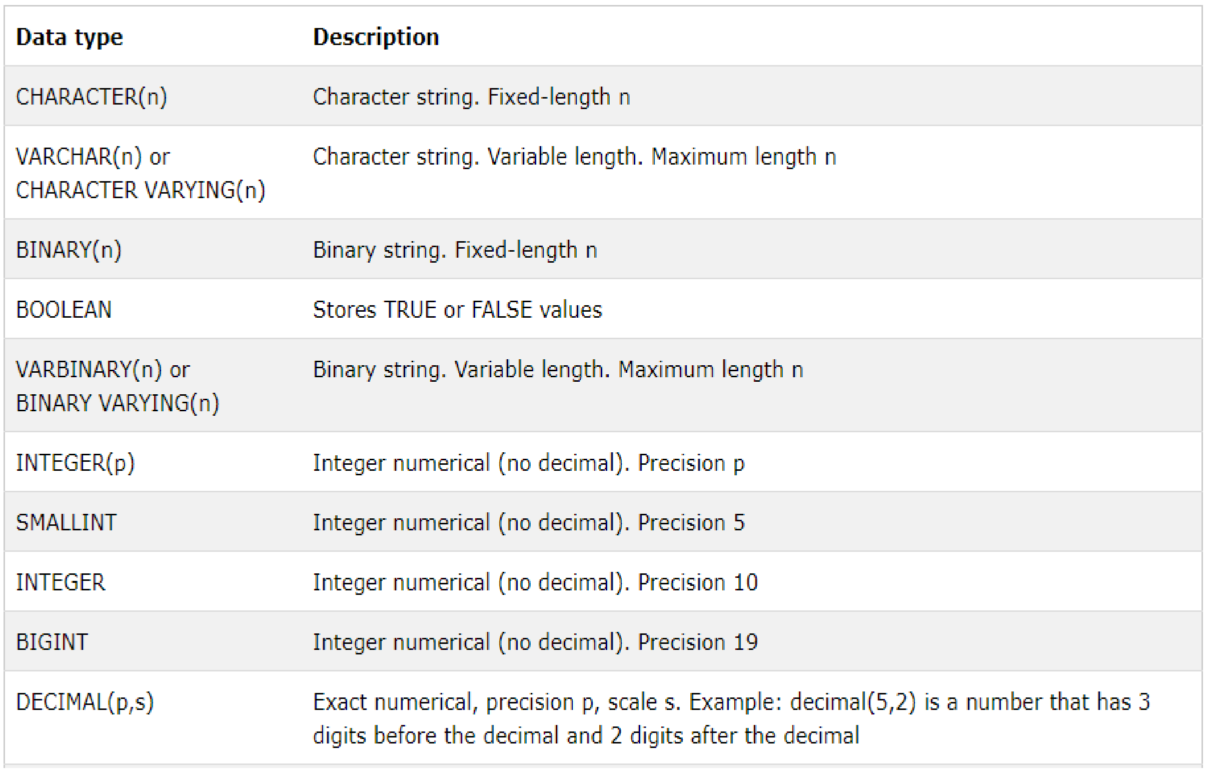
Π author (Books) ∪ Π author (Articles)
Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−) **[Any relevant example]**
The result of set difference operation is tuples, which are present in one relation but are not in the second relation.
Notation − (r − s)
Finds all the tuples that are present in r but not in s.
Π author (Books) − Π author (Articles)
Output
Provides the name of authors who have written books but not articles.

Rename Operation (ρ)
The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.
Notation − ρ x (E)

b) List various data types of SQL and explain. 4M

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has a related data type in SQL. You can use these data types while creating your tables. You can choose a data type for a table column based on your requirement.



Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19
DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal

(OR)

5.) a. Write short notes on Domain Relational Calculus. 6M

[Domain relational calculus Explanation--4M]
[Example--2M]

There is another type of relational calculus called the domain relational calculus, or simply, domain calculus. Domain calculus differs from tuple calculus in the type of variables used in formulas: Rather than having variables range over tuples, the variables range over single values from domains of attributes. To form a relation of degree n for a query result, we must have n of these domain variables—one for each attribute. An expression of the domain calculus is of the form {x1, x2, ..., xn | COND(x1, x2, ..., xn, xn+1, xn+2, ..., xn+m)}
where x1, x2, ..., xn, xn+1, xn+2, ..., xn+m are domain variables that range over domains (of attributes), and COND is a condition or formula of the domain relational calculus.
A formula is made up of atoms. The atoms of a formula are slightly different from those for the tuple calculus and can be one of the following:
1. An atom of the form R(x1, x2, ..., xj), where R is the name of a relation of degree j and each xi, $1 \leq i \leq j$, is a domain variable. This atom states that a list of values of <x1, x2, ..., xj> must be a tuple in the relation whose name is R, where xi is the value of the ith attribute value of the tuple. To make a domain calculus expression more concise, we can drop the commas in a list of variables; thus, we can write:
{x1, x2, ..., xn | R(x1 x2 x3) AND ...} instead of:
{x1, x2, ... , xn | R(x1, x2, x3) AND ...}
2. An atom of the form xi op xj, where op is one of the comparison operators in the set {=, <, ≤, >, ≥, ≠}, and xi and xj are domain variables.
3. An atom of the form xi op c or c op xj, where op is one of the comparison operators in the set {=, <, ≤, >, ≥, ≠}, xi and xj are domain variables, and c is a constant value.

b) Discuss various types of Constraints in SQL. 6M

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

UNIT III

6. a) Write the algorithms for insertion and deletion of an element in B+ Tree. 4M

B+ Tree Insertion

B+ trees are filled from bottom and each entry is done at the leaf node.

If a leaf node overflows −

Split node into two parts.
Partition at **i = ⌊(m+1)/2⌋.**
First **i** entries are stored in one node.
Rest of the entries (i+1 onwards) are moved to a new node.
ith key is duplicated at the parent of the leaf.

If a non-leaf node overflows −

Split node into two parts.
Partition the node at i = ⌈ (m+1)/2⌉ .
Entries up to i are kept in one node.
Rest of the entries are moved to a new node.

B+ Tree Deletion

B+ tree entries are deleted at the leaf nodes.

The target entry is searched and deleted.

If it is an internal node, delete and replace with the entry from the left position.

After deletion, underflow is tested,

If underflow occurs, distribute the entries from the nodes left to it.
If distribution is not possible from left, then
Distribute from the nodes right to it.
If distribution is not possible from left or from right, then
Merge the node with left and right to it.

b) Compare BCNF and 3NF with an example. 8M

Difference between 3NF and BCNF Explanation --3M

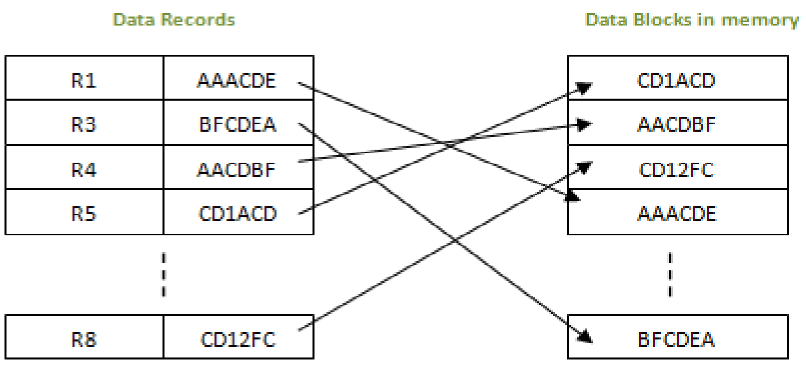
Both 3NF and BCNF are normal forms that are used in relational databases to minimize redundancies in tables. In a table that is in the BCNF normal form, for every non-trivial functional dependency of the form A → B, A is a super-key whereas, a table that complies with 3NF should be in the 2NF, and every non-prime attribute should directly depend on every candidate key of that table. BCNF is considered as a stronger normal form than the 3NF and it was developed to capture some of the anomalies that could not be captured by 3NF.

Obtaining a table that complies with the BCNF form will require decomposing a table that is in the 3NF. This decomposition will result in additional join operations (or Cartesian products) when executing queries. This will increase the computational time. On the other hand, the tables that comply with BCNF would have fewer redundancies than tables that only comply with 3NF. Furthermore, most of the time, it is possible to obtain a table that comply with 3NF without hindering dependency preservation and lossless joining. But this is not always possible with BCNF.

(OR)

7. a) Discuss about Indexed Sequential Access Methods(ISAM) with neat sketches. 8M

This is an advanced sequential file organization method. Here records are stored in order of primary key in the file. Using the primary key, the records are sorted. For each primary key, an index value is generated and mapped with the record. This index is nothing but the address of record in the file.



In this method, if any record has to be retrieved, based on its index value, the data block address is fetched and the record is retrieved from memory.

Advantages of ISAM

- Since each record has its data block address, searching for a record in larger database is easy and quick. There is no extra effort to search records. But proper primary key has to be selected to make ISAM efficient.
- This method gives flexibility of using any column as key field and index will be generated based on that. In addition to the primary key and its index, we can have index generated for other fields too.
- Hence searching becomes more efficient, if there is search based on columns other than primary key.
- It supports range retrieval, partial retrieval of records. Since the index is based on the key value, we can retrieve the data for the given range of values. In the same way, when a partial key value is provided, say student names starting with 'JA' can also searched easily.

Disadvantages of ISAM

- An extra cost to maintain index has to be afforded. i.e.; we need to have extra space in the disk to store this index value. When there is multiple key-index combinations, the disk space will also increase.
- As the new records are inserted, these files have to be restructured to maintain the sequence. Similarly, when the record is deleted, the space used by it needs to be released. Else, the performance of the database will slow down.

b) Describe Join Dependency with an example. 4M

Definition. A **join dependency** (**JD**), denoted by JD(*R*1, *R*2, ..., *Rn*), specified on relation schema *R*, specifies a constraint on the states *r* of *R*. The constraint states that every legal state *r* of *R* should have a nonadditive join decomposition into *R*1, *R*2, ..., *Rn*. Hence, for every such *r* we have

$$* (\pi_{R_1}(r), \pi_{R_2}(r), ..., \pi_{R_n}(r)) = r$$

Notice that an MVD is a special case of a JD where n = 2. That is, a JD denoted as JD(R1, R2) implies an MVD $(R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$ (or, by symmetry, (*R*1 ∩ *R*2)->>(*R*2 – *R*1)). A join dependency JD(*R*1, *R*2, ..., *Rn*), specified on relation schema *R*, is a **trivial** JD if one of the relation schemas *Ri* in JD(*R*1, *R*2, ..., *Rn*) is equal to *R*. Such a dependency is called trivial because it has the nonadditive join property for any relation state *r* of *R* and thus does not specify any constraint on *R*.We can now define fifth normal form, which is also called *project-join normal form.*

Definition. A relation schema *R* is in **fifth normal form (5NF)** (or **project-joinnormal form (PJNF)**) with respect to a set *F* of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(*R*1, *R*2, ..., *Rn*) in *F*+ (that is, implied by *F*),18 every *Ri* is a superkey of *R*.

UNIT IV

8. a) What are the desirable properties of a transaction? 4M

Transactions should possess several properties, often called the ACID properties; they should be enforced by the concurrency control and recovery methods of the DBMS. The following are the ACID properties:

■ **Atomicity.** A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

■ **Consistency preservation.** A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

■ **Isolation.** A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing. concurrently. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

■ **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.

b) Write short notes on Shadow Paging. 8M

Shadow paging is an alternative to log-based recovery techniques, which has both advantages and disadvantages. It may require fewer disk accesses, but it is hard to extend paging to allow multiple concurrent transactions. The paging is very similar to paging schemes used by the operating system for memory management. The idea is to maintain two page tables during the life of a transaction: the current page table and the shadow page table. When the transaction starts, both tables are identical. The shadow page is never changed during the life of the transaction. The current page is updated with each write operation. Each table entry points to a page on the disk. When the transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released.
If the shadow is stored in nonvolatile memory and a system crash occurs, then the shadow page table is copied to the current page table.

This guarantees that the shadow page table will point to the database pages corresponding to the state of the database prior to any transaction that was active at the time of the crash, making aborts automatic.

(OR)

9. a) Discuss about Two-Phase locking techniques for Concurrency Control. 6M

Two-phase locking techniques---6M
Time stamp ordering---6M

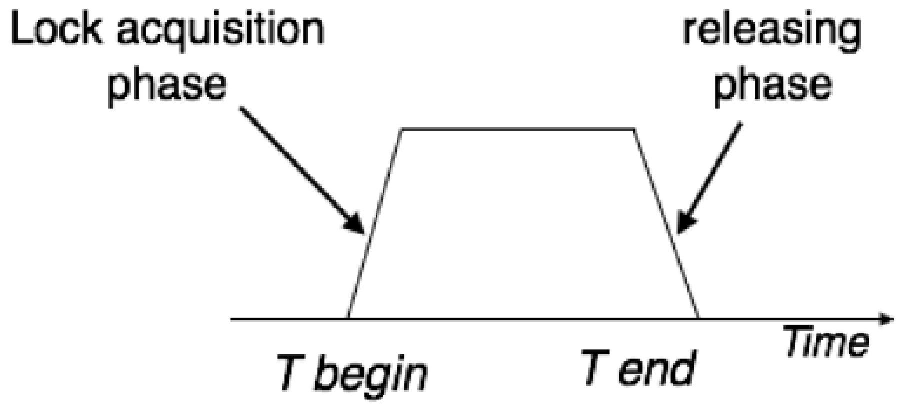
In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols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

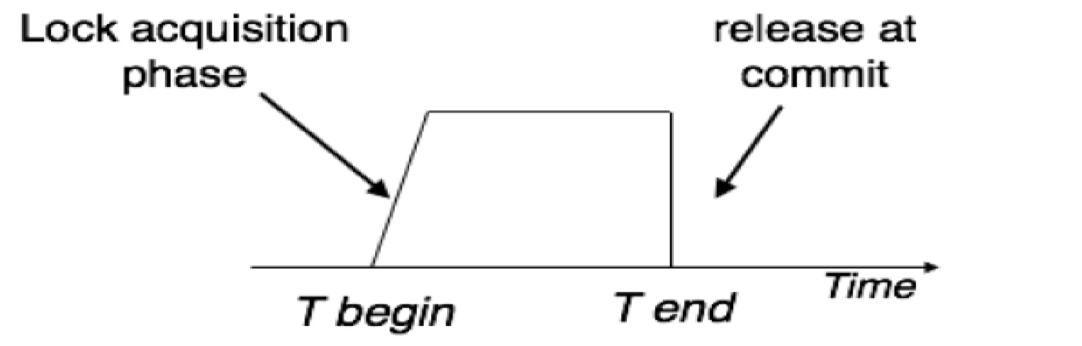
Two-Phase Locking 2PL
 This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction 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.



Strict-2PL does not have cascading abort as 2PL does.

b) Discuss about Recovery techniques based on Deferred Update. 6M

Recovery Techniques Based on Deferred Update [Explanation--6M]

These techniques defer or postpone any actual updates to the database until the transaction reaches it commit point. During transaction execution, the updates are written to the log file. After the transaction reaches it commit point, the log file is force-written to disk, then the updates are recorded in the database. If the transaction fails before reaching its commit point, there is no need to undo any operations because the transaction has not affected the database on disk in any way.

A typical deferred update protocol uses the following procedure:

A transaction cannot change the database on disk until it reaches its commit point. A transaction does not reach its commit point until all its update operations are recorded in the log file and the log file is force-written to disk. Recovery techniques based on deferred update are therefore known as NO UNDO/REDO techniques. REDO is needed in case the system fails after a transaction commits but before all its changes are recorded on disk. In this case, the transaction operations are redone from the log file.

Recovery Using Deferred Update in a Single-User Environment

RDU_S (**R**ecovery using **D**eferred **U**pdate in a **S**ingle-User environment) uses A REDO procedure as follows:

PROCEDURE RDU_S:

- Use two lists of transactions: the committed transactions since the last checkpoint, and the active transactions (at most one because the system is single-user).

- Apply the following REDO operation to all the WRITE_ITEM operations of the committed transactions and restart the active transactions:

REDO(WRITE_OP): Redoing a write_item operation WRITE_OP consists of examining its log entry [write_item,T,X,old_value,new_value] and setting the value of item X in the database to its new_value, which is the after image (AFIM).

Deferred Update with Concurrent Execution in a Multiuser Environment

RDU_M(**R**ecovery Using **D**eferred **U**pdate in a **M**ultiuser environment) algorithm is as follows where the REDO procedure is as defined above:

PROCEDURE RDU_M:

- Use two lists of truncations:

- T: committed transactions since the last checkpoint (commit list).
- T: active transactions (active list). '
- REDO all the WRITE operations of the committed transactions from the log file, in the order in which they were written in the log.

- The transactions that are active and did not commit are cancelled and must be resubmitted.

The previous algorithm can be made more efficient by noting that if a database item X has been updated more than once by committed transactions since the last checkpoint, it is only necessary to REDO the last update of X from the log file during recovery as the other updates would be overwritten by the last update anyway.

To implement this, start from the end of the log; then whenever an item is redone, it is added to the list of redone items. Before a REDO is applied to an item, the list is checked; if the item appears on the list, it is not redone again, since its most updated value has already been recovered.

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