14CS IT 504

III/IV B.Tech (Supplementary) DEGREE EXAMINATION

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

Fifth Semester **DATABASE MANAGEMENT SYSYEMS**

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=12Marks)

a) What is Data Independence?

Data independence is ability to modify a schema definition in one level without affecting a schema definition in the next higher level.
There are two levels of data independence:
Physical Data Independence
Logical Data Independence

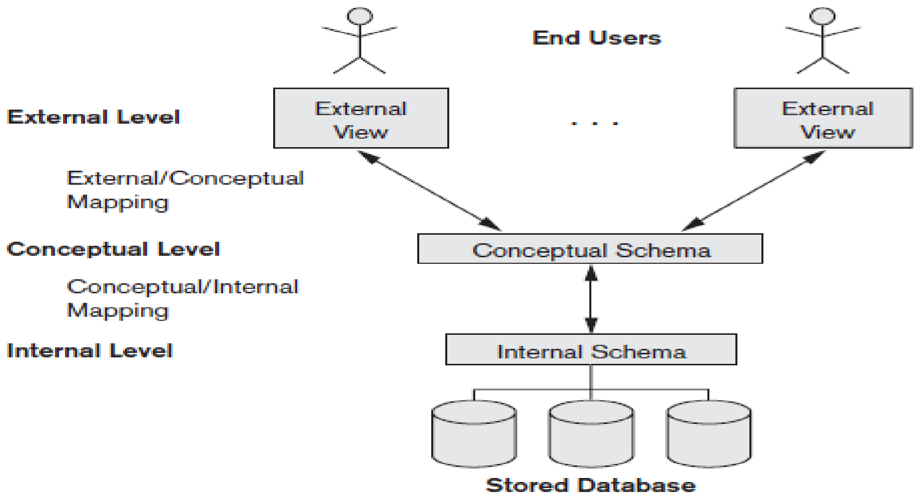
b) Write any three Relationship types.

- Unary Relationship
  An ENTITY TYPE linked with itself, also called recursive relationship
- Binary relationship
  A Binary relationship is the one that links two entities sets
- Ternary Relationship
  A Ternary relationship is the one that involves three entities

c) What is Strong entity and Weak entity?

The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set. An entity set that has a primary key is called as Strong entity set

d) Draw three schema architecture.



e) What are the DML & DCL Commands?

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:
CREATE, ALTER ,DROP, TRUNCATE,RENAME

f) Define view

A view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database

g) Differentiate relationship instance and relationship type.

Relationship instance is an association of entities, where the association includes exactly one entity from each participating entity type.

Relationship type defines a relationship set among entities from these types.

h) Name the binary relational operations.

Join and Division

i) Define atomicity and durability.

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

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.

j) Define shadow paging.

Shadow paging is a copy-on-write technique for avoiding in-place updates of pages. Instead, when a page is to be modified, a **shadow page** is allocated. Since the shadow page has no references (from other pages on disk), it can be modified liberally, without concern for consistency constraints,

k) Define Recoverability.

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason

l) What is meant by Multiple-Granularity?

Multiple granularity locking (MGL) is a locking method used in database management systems (DBMS) and relational databases. In MGL, locks are set on objects that contain other objects. MGL exploits the hierarchical nature of the contains relationship.

UNIT I

2. Discuss about ER Models and how do you refine the ER design for a Company Database Using ER diagrams, Naming conventions and some design issues. 12M

Explanation--6M

In our example, we specify the following relationship types:

■ MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation.The attribute Start_date is assigned to this relationship type.

■ WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.

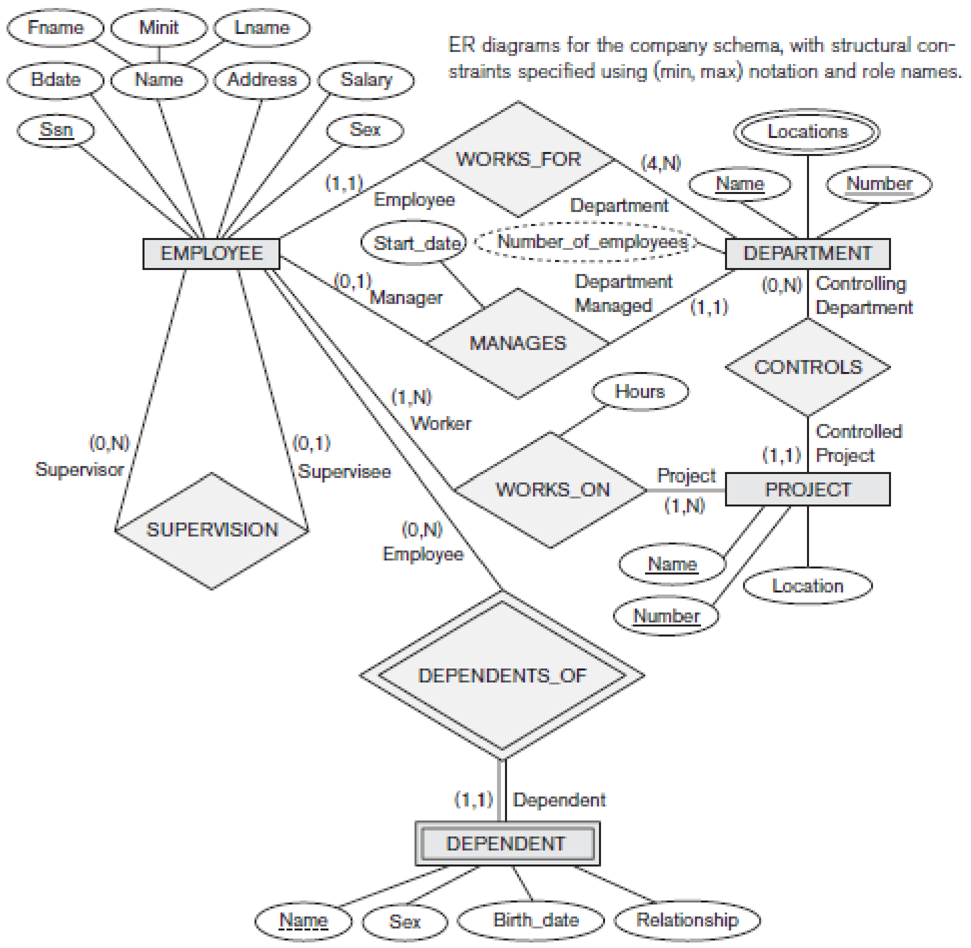
■ CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.

■ SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.

■ WORKS_ON, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.

■ DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

Diagram ---6M



(OR)

3. a) Analyze classification of Database Management systems. 6M

Several criteria are normally used to classify DBMSs. The first is the data model on which the DBMS is based. The main data model used in many current commercial DBMSs is the relational data model. The object data model has been implemented in some commercial systems but has not had widespread use.Many legacy applications still run on database systems based on the hierarchical and network data models. Examples of hierarchical DBMSs include IMS (IBM) and some other systems like System 2K (SAS Inc.) and TDMS. IMS is still used at governmental and industrial installations, including hospitals and banks, although many of its users have converted to relational systems. The relational DBMSs are evolving continuously, and, in particular, have been incorporating many of the concepts that were developed in object databases. This has led to a new class of DBMSs called object-relational DBMSs.We can categorize DBMSs based on the data model: relational, object, object-relational, hierarchical, network, and other.

More recently, some experimental DBMSs are based on the XML (eXtended Markup Language) model, which is a tree-structured (hierarchical) data model. These have been called native XML DBMSs. Several commercial relational DBMSs have added XML interfaces and storage to their products.

The second criterion used to classify DBMSs is the number of users supported by the system. Single-user systems support only one user at a time and are mostly used with PCs. Multiuser systems, which include the majority of DBMSs, support concurrent multiple users.

The third criterion is the number of sites over which the database is distributed. A DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. Homogeneous DDBMSs use the same DBMS software at all the sites, whereas heterogeneous DDBMSs can use different DBMS software at each site.

b) List & explain characteristics and responsibilities of Data models. 6M

The role of data models Explanation--3M

The main aim of data models is to support the development of information systems by providing the definition and format of data. According to West and Fowler (1999) "if this is done consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data. The results of this are indicated above. However, systems and interfaces often cost more than they should, to build, operate, and maintain. They may also constrain the business rather than support it. A major cause is that the quality of the data models implemented in systems and interfaces is poor".

- Business rules, specific to how things are done in a particular place, are often fixed in the structure of a data model. This means that small changes in the way business is conducted lead to large changes in computer systems and interfaces.
- Entity types are often not identified, or incorrectly identified. This can lead to replication of data, data structure, and functionality, together with the attendant costs of that duplication in development and maintenance.
- Data models for different systems are arbitrarily different. The result of this is that complex interfaces are required between systems that share data. These interfaces can account for between 25-70% of the cost of current systems.
- Data cannot be shared electronically with customers and suppliers, because the structure and meaning of data has not been standardized.

Explain Characteristics ----3M

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.

UNIT II

4. Differentiate Tuple Relational Calculus with Domain Relational Calculus with Examples. 12M

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

The tuple relational calculus is based on specifying a number of tuple variables. Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation. A simple tuple relational calculus query is of the form:
{t | COND(t)}
where t is a tuple variable and COND(t) is a conditional (Boolean) expression involving t that evaluates to either TRUE or FALSE for different assignments of tuples to the variable t. The result of such a query is the set of all tuples t that evaluate COND(t) to TRUE. These tuples are said to satisfy COND(t). For example, to find all employees whose salary is above $50,000, we can write the following tuple calculus expression:
{t | EMPLOYEE(t) AND t.Salary>50000}
The condition EMPLOYEE(t) specifies that the range relation of tuple variable t is EMPLOYEE. Each EMPLOYEE tuple t that satisfies the condition t.Salary>50000 will be retrieved.
The above query retrieves all attribute values for each selected EMPLOYEE tuple t. To retrieve only some of the attributes—say, the first and last names—we write
{t.Fname, t.Lname | EMPLOYEE(t) AND t.Salary>50000}
Informally, we need to specify the following information in a tuple relational calculus expression:
■ For each tuple variable t, the range relation R of t. This value is specified by a condition of the form R(t). If we do not specify a range relation, then the variable t will range over all possible tuples "in the universe" as it is not restricted to any one relation.
■ A condition to select particular combinations of tuples. As tuple variables range over their respective range relations, the condition is evaluated for every possible combination of tuples to identify the selected combinations for which the condition evaluates to TRUE.
■ A set of attributes to be retrieved, the requested attributes. The values of these attributes are retrieved for each selected combination of tuples.

[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.

(OR)

5. a) Consider the relational schema R = {E,F,G,H,I,J,K,L,M,N} and set Functional dependencies {{E,F}→{G},{F}→{I,J},{E,H}→{K,L},{K}→{M},{L}→{N}} On R. What are the candidate keys for R. 8M

Definition of Super key---2M
Definition of Candidate key--2M
Problem---4M

The **set of attributes** whose attribute closure is set of all attributes of relation is called super key of relation. The **minimal set of attributes** whose attribute closure is set of all attributes of relation is called candidate key of relation

Identify the closure of LHS of FD

$\{F\}^+ \rightarrow \{F,I,J\}$ [Not determine all attributes of R]
$\{K\}^+ \rightarrow \{M,K\}$ [Not determine all attributes of R]
$\{L\}^+ \rightarrow \{N,L\}$ [Not determine all attributes of R]
$\{E,F\}^+ \rightarrow \{E,F,G,I,J\}$ [Not determine all attributes of R]
$\{E,H\}^+ \rightarrow \{K,L,E,H,M,N\}$ [Not determine all attributes of R]

If no super key found from step 1 , then follow step 2 to find a new key

$\{E,H,K\}^+ \rightarrow \{K,L,E,H,M,N\}$ [Not determine all attributes of R]
$\{E,F,H\}^+ \rightarrow \{E,F,G,I,J,K,L,H,M,N\}$ [Determine all attributes of R]

From the above steps minimal set of attributes whose attribute closure is set of all attributes of relation is called candidate key. Hence {E,F,H} is a candidate key or a minimal set of super key.

b) Explain about JOIN and DIVISION operations with Examples. 4M

The JOIN Operation [Explanation of Join operation with example--2M]

The JOIN operation, denoted by , is used to combine related tuples from two relations into single "longer" tuples. This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations. To get the manager's name, we need to combine each department tuple with the employee tuple whose Ssn value matches the Mgr_ssn value in the department tuple.We do this by using the JOIN operation and then projecting the result over the necessary attributes, as follows:

Any relevant example can be considered

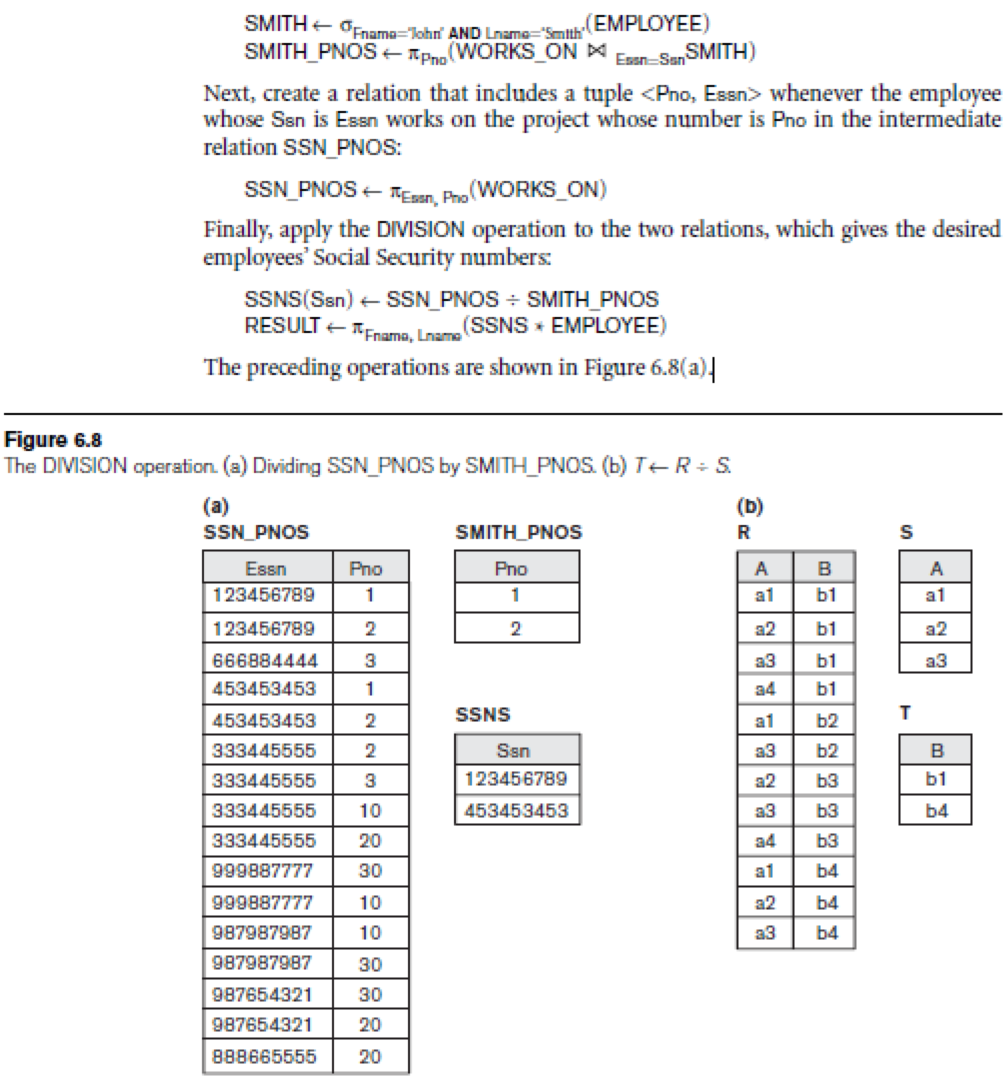
$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$
$\text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})$

The first operation is illustrated in Figure. Note that Mgr_ssn is a foreign key of the DEPARTMENT relation that references Ssn, the primary key of the EMPLOYEE relation. This referential integrity constraint plays a role in having matching tuples in the referenced relation EMPLOYEE

The DIVISION Operation **[Explanation of Division operation with example--2M]**

The DIVISION operation, denoted by ÷, is useful for a special kind of query that sometimes occurs in database applications. An example is Retrieve the names of employees who work on all the projects that 'John Smith' works on. To express this query using the DIVISION operation, proceed as follows. First, retrieve the list of project numbers that 'John Smith' works on in the intermediate relation SMITH_PNOS:

Any relevant example can be considered



$\text{SMITH} \leftarrow \sigma_{\text{Fname='John' AND Lname='Smith'}}(\text{EMPLOYEE})$
$\text{SMITH\_PNOS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} \bowtie_{\text{Essn=Ssn}} \text{SMITH})$

Next, create a relation that includes a tuple <Pno, Essn> whenever the employee whose Ssn is Essn works on the project whose number is Pno in the intermediate relation SSN_PNOS:

$\text{SSN\_PNOS} \leftarrow \pi_{\text{Essn, Pno}}(\text{WORKS\_ON})$

Finally, apply the DIVISION operation to the two relations, which gives the desired employees' Social Security numbers:

$\text{SSNS(Ssn)} \leftarrow \text{SSN\_PNOS} \div \text{SMITH\_PNOS}$
$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}}(\text{SSNS} * \text{EMPLOYEE})$

The preceding operations are shown in Figure 6.8(a).

Figure 6.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

UNIT III

6. a) Discuss about the Dynamic Multilevel Indexes. 6M

B-trees and B+-trees are special cases of the well-known search data structure known as a tree.We briefly introduce the terminology used in discussing tree data structures. A tree is formed of nodes. Each node in the tree, except for a special node called the root, has one parent node and zero or more child nodes. The root node has no parent. A node that does not have any child nodes is called a leaf node; a nonleaf node is called an internal node. The level of a node is always one more than the level of its parent, with the level of the root node being zero.5 A subtree of a node consists of that node and all its descendant nodes—its child nodes, the child nodes of its child nodes, and so on.

Explanation of B-Trees ---3M

B-Trees. The B-tree has additional constraints that ensure that the tree is always balanced and that the space wasted by deletion, if any, never becomes excessive. The algorithms for insertion and deletion, though, become more complex in order to maintain these constraints. Nonetheless, most insertions and deletions are simple processes; they become complicated only under special circumstances—namely, whenever we attempt an insertion into a node that is already full or a deletion from a node that makes it less than half full. More formally, a B-tree of order p, when used as an access structure on a key field to search for records in a data file, can be defined as follows:

1. Each internal node in the B-tree is of the form
<P1, <K1, Pr1>, P2, <K2, Pr2>, ..., <Kq–1, Prq–1>, Pq>
where q ≤ p. Each Pi is a tree pointer—a pointer to another node in the Btree. Each Pri is a data pointer8—a pointer to the record whose search key field value is equal to Ki (or to the data file block containing that record).
2. Within each node, K1 < K2 < ... < Kq−1.
3. For all search key field values X in the subtree pointed at by Pi we have:
Ki–1 < X < Ki for 1 < i < q; X < Ki for i = 1; and Ki–1 < X for i = q.
4. Each node has at most p tree pointers.
5. Each node, except the root and leaf nodes, has at least $\lceil (p/2) \rceil$ tree pointers.
The root node has at least two tree pointers unless it is the only node in the tree.
6. A node with q tree pointers, q ≤ p, has q – 1 search key field values (and hence has q – 1 data pointers).
7. All leaf nodes are at the same level. Leaf nodes have the same structure as internal nodes except that all of their tree pointers Pi are NULL

B+ Trees **Explanation of B+-Trees ---3M**

Most implementations of a dynamic multilevel index use a variation of the B-tree data structure called a B+-tree. In a B-tree, every value of the search field appears once at some level in the tree, along with a data pointer. In a B+-tree, data pointers are stored only at the leaf nodes of the tree; hence, the structure of leaf nodes differs from the structure of internal nodes. The leaf nodes have an entry for every value ofthe search field, along with a data pointer to the record (or to the block that contains this record) if the search field is a key field. For a nonkey search field, the pointer points to a block containing pointers to the data file records, creating an extra level of indirection.

The leaf nodes of the B+-tree are usually linked to provide ordered access on the search field to the records. These leaf nodes are similar to the first (base) level of an index. Internal nodes of the B+-tree correspond to the other levels of a multilevel index. Some search field values from the leaf nodes are repeated in the internal nodes of the B+-tree to guide the search. The structure of the internal nodes of a B+ tree of order p is as follows:

1. Each internal node is of the form
   $<P_1, K_1, P_2, K_2, ..., P_{q-1}, K_{q-1}, P_q>$
   where $q \leq p$ and each P_i is a **tree pointer.**
2. Within each internal node, $K_1 < K_2 < ... < K_{q-1}$.
3. For all search field values X in the subtree pointed at by P_i, we have $K_{i-1} < X \leq K_i$ for $1 < i < q$; $X \leq K_i$ for $i = 1$; and $K_{i-1} < X$ for $i = q$
4. Each internal node has at most p tree pointers.
5. Each internal node, except the root, has at least $\lceil (p/2) \rceil$ tree pointers. The root node has at least two tree pointers if it is an internal node.
6. An internal node with q pointers, $q \leq p$, has $q - 1$ search field values.

b) Give definitions for all Normal Forms and differentiate 3NF & BCNF. 6M

Definition for all Normal forms---3M

First Normal Form

A relation R is in first normal form (1NF) if and only if all underlying domains contains atomic values only.

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 dependent on the primary key.

Third Normal Form

A relation R is in third normal form (3NF) if and only if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key.

Boyce/Codd Normal Form

A relation R is in Boyce/Codd normal form (BCNF) if and only if every determinant is a candidate key.

Fourth Normal Form

A relation R is in fourth normal form (4NF) if and only if, wherever there exists an MVD in R, say A -> -> B, then all attributes of R are also functionally dependent on A. In other words, the only dependencies (FDs or MVDs) in R are of the form K -> X (i.e. a functional dependency from a candidate key K to some other attribute X). Equivalently: R is in 4NF if it is in BCNF and all MVD's in R are in fact FDs.

Fifth Normal Form

A relation R is in fifth normal form (5NF) – also called **projection-join** normal form (PJ/NF) if and only if every join dependency in R is a consequence of the candidate keys of R.

For every normal form it is assumed that every occurrence of R can be uniquely identified by a primary key using one or more attributes in R.

FD = Functional Dependency
MVD = Multi-Valued Dependency

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) Describe about Relational database schema design. **6M**

Definition ----2M
Example Explanation--2M
Any relevant example--2M

A relational database usually contains many relations, with tuples in relations that are related in various ways. A relational database schema S is a set of relation schemas S = {R1, R2, ..., Rm} and a set of integrity constraints IC. A relational database state10 DB of S is a set of relation states DB = {r1, r2, ..., rm} such that each ri is a state of Ri and such that the ri relation states satisfy the integrity constraints specified in IC.

A relational database schema S is a set of relation schemas S = {R1, R2, ..., Rm} and a set of integrity constraints IC. A relational database state10 DB of S is a set of relation states DB = {r1, r2, ..., rm} such that each ri is a state of Ri and such that the ri relation states satisfy the integrity constraints specified in IC. Figure shows a relational database schema that we call COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT}. When we refer to a relational database, we implicitly include both its schema and its current state. A database state that does not obey all the integrity constraints is called an invalid state, and a state that satisfies all the constraints in the defined set of integrity constraints IC is called a valid state.

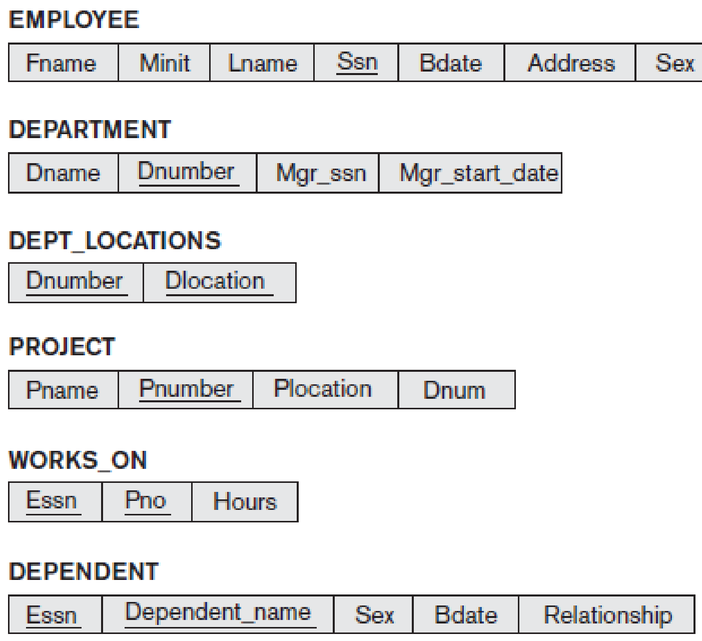

EMPLOYEE
Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno
DEPARTMENT
Dname | Dnumber | Mgr_ssn | Mgr_start_date
DEPT_LOCATIONS
Dnumber | Dlocation
PROJECT
Pname | Pnumber | Plocation | Dnum
WORKS_ON
Essn | Pno | Hours
DEPENDENT
Essn | Dependent_name | Sex | Bdate | Relationship

Figure
Schema diagram for the COMPANY relational database schema.

b) Demonstrate operations on files. 6M

Actual operations for locating and accessing file records vary from system to system. Below, we present a set of representative operations. Typically, high-level programs, such as DBMS software programs, access records by using these commands, so we sometimes refer to program variables in the following descriptions:

■ Open. Prepares the file for reading or writing. Allocates appropriate buffers (typically at least two) to hold file blocks from disk, and retrieves the file header. Sets the file pointer to the beginning of the file.

■ Reset. Sets the file pointer of an open file to the beginning of the file.

■ Find (or Locate). Searches for the first record that satisfies a search condition. Transfers the block containing that record into a main memory buffer (if it is not already there). The file pointer points to the record in the buffer and it becomes the current record. Sometimes, different verbs are used to indicate whether the located record is to be retrieved or updated.

■ Read (or Get). Copies the current record from the buffer to a program variable in the user program. This command may also advance the current record pointer to the next record in the file, which may necessitate reading the next file block from disk.

■ FindNext. Searches for the next record in the file that satisfies the search condition. Transfers the block containing that record into a main memory buffer (if it is not already there). The record is located in the buffer and becomes the current record. Various forms of FindNext (for example, Find Next record within a current parent record, Find Next record of a given type, or Find Next record where a complex condition is met) are available in legacy DBMSs based on the hierarchical and network models.

■ Delete. Deletes the current record and (eventually) updates the file on disk to reflect the deletion.

■ Modify. Modifies some field values for the current record and (eventually) updates the file on disk to reflect the modification.

■ Insert. Inserts a new record in the file by locating the block where the record is to be inserted, transferring that block into a main memory buffer (if it is not already there), writing the record into the buffer, and (eventually) writing the buffer to disk to reflect the insertion.

■ Close. Completes the file access by releasing the buffers and performing any other needed cleanup operations.

UNIT IV

8. a) How do you characterize schedules based on Recoverability. 7M

Recoverability Definition---2M
Explanation--5M

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) State and Explain desirable properties in transaction processing. 5M

Definition--2M

ACID Properties---3M

A transaction is typically implemented by a computer program, which includes database commands such as retrievals, insertions, deletions, and updates.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.

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.

(OR)

9. Explain any two Database Recovery techniques in detail with examples. 12M

There are two main techniques for recovery:

1) Deferred Update: transaction updates are not written to disk until after a transaction reaches its commit point.
2) Immediate Update: transaction updates are immediately written to disk. If a transaction fails after recording some changes in the database but before reaching its commit point, the effect of its operations on the database must be undone (rolledback).

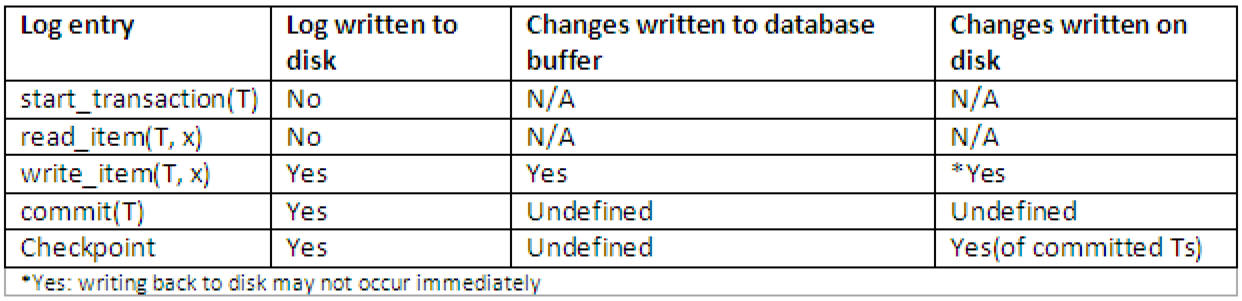
Recovery Techniques Based on Immediate Update **[Explanation--6M]**

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 10.7 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

.

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**

Paper Evaluators:

Sno	**Name of the college**	**Name of the examiner**	**Signature**